The system includes at least one connector (1) for coupling the external memory to the host processing system. An external memory (10) stores two sets of program instructions for the videographics program for execution by the host processing unit.

A graphics processor (2) is coupled to the external memory (10) and coupled, in use, to the host processin unit via the connector, for executing the second set of instructions.

The host unit is formed by a video game system main processing unit, whilst the external memory is a game cartridge.

USE/ADVANTAGE - Image processing appts. Video game system. Fully user programmable graphics coprocessor. Host processor has constant access to 16 general registers of graphics coprocessor.

Dwg.1/21

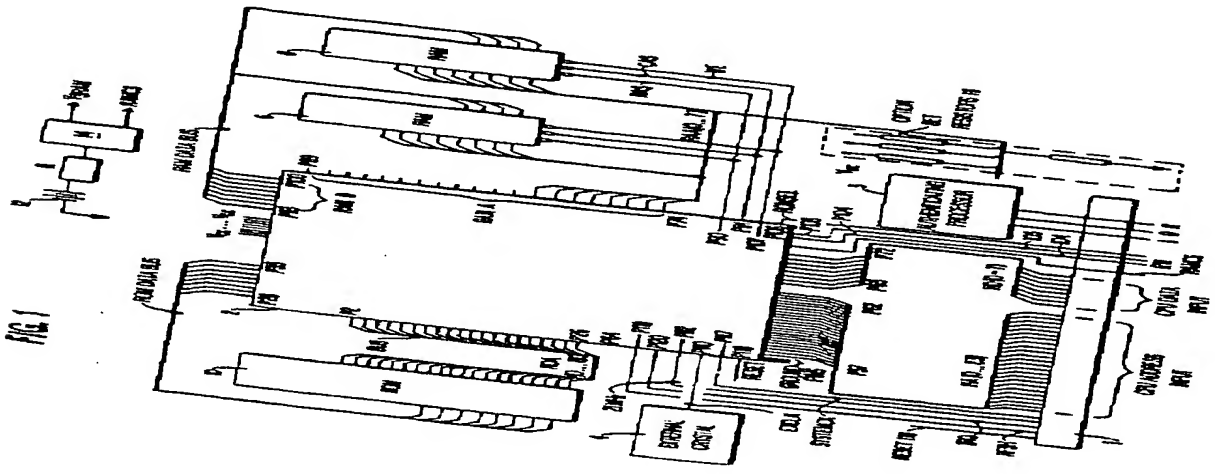
US 5388841 A

The fully programmable, graphics microprocessor is embodied in a removable external memory unit for connection with a host information processing system. A video game system includes a host video game system and a pluggable video game cartridge housing the graphics microprocessor. The game cartridge also includes a read-only program memory (ROM) and a random-access memory (RAM). The graphics coprocessor operates in conjunction with a three bus architecture embodied on the game cartridge. The graphics processor using this bus architecture may execute programs from either the program ROM, external RAM or its own internal cache RAM.

The fully user programmable graphics coprocessor has an instruction set which is designed to efficiently implement arithmetic operations associated with 3-D graphics and, for example, includes special instructions executed by dedicated hardware for plotting individual pixels in the host video game system's character mapped display which, from the programmer's point of view, creates a "virtual" bit map by permitting the addressing of individual pixels-even though the host system is character based. The graphics coprocessor interacts with the host coprocessor such that the graphics coprocessor's 16 general registers are accessible to the host processor at all times.

ADVANTAGE - Enables video game system to do high speed graphics processing.

Dwg.1/21



(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(51) Int. Cl.[6]
G06T 1/00

(45) 공고일자 2001년02월01일
(11) 등록번호 10-0280939
(24) 등록일자 2000년11월14일

(21) 출원번호 10-1992-0018673
(22) 출원일자 1992년10월10일
(65) 공개번호 특1993-0016902
(43) 공개일자 1993년08월30일

(30) 우선권주장 7/827,098
1992년01월30일
미국(US)

(73) 특허권자 에이/엔 인코퍼레이티드 아라카와 미노루
미합중국 워싱턴 레드몬드 사서함 957 엔. 이. 제 150 애버뉴 4820

(72) 발명자 제레미 이. 산
영국 런던 밀힐 업힐로드 73

벤 치이즈
영국 하트퍼드셔 멜버른로이스톤 돌핀레인 23

칼 엔. 그레이엄
영국 런던 엔. 칭포드 우드랜드로드 15

피터 알. 워네스
영국 런던 이스트함 월풀로드 12A

(74) 대리인 김명신

심사관 : 신상호

(54) 텔레비젼형 디스플레이를 사용하는 비디오게임시스템

요약

주 정보처리시스템과 연결된 이동식 외부메모리유닛에 내장되도록 설계되어 전적으로 프로그램가능한 그래픽프로세서가 공지된다.

한 예시적실시예에서, 주비디오게임시스템 및, 그래픽마이크로프로세서를 수용하고 있는 플러그접속가능한 비디오게임카트리지를 포함하는 비디오 게임시스템이 기술된다.

또한 상기 게임카트리지는 ROM과 RAM을 포함하고 있다.

그래픽코프로세서는 게임카트리지에 내장된 3개의 버스구조를 이용한다.

이 3버스구조를 사용하는 그래픽프로세서는 프로그램 ROM, 외부 RAM 또는 자기의 내부캐시 RAM중 하나에 기억되어 있는 프로그램을 실행한다.

전적으로 유저(사용자)에 의해 프로그램가능한 그래픽코프로세서는 3차원 그래픽과 관련된 산술동작을 효과적으로 실행하도록 설계된 명령세트, 예를들면 주비디오게임시스템의 문자맵핑 디스플레이에 개별픽셀을 플로팅하기 위해 전용하드웨어에 의해 실행되는 특수명령을 포함하는 명령세트를 포함하고 있다.

그래픽코프로세서는 주코프로세서와 상호작용하기 때문에 그래픽코프로세서의 16개의 범용레지스터들은 항상 주코프로세서에 의해 엑세스될 수 있다.

[발명의 명칭]

텔레비젼형 디스플레이를 사용하는 비디오게임시스템

[도면의 간단한 설명]

제1도는 본 발명의 예시적 실시예에 따른 외부메모리시스템의 블록다이어그램.

제2도는 본 발명의 예시적 실시예의 그래픽 코프로세서와 함께 사용하기 위한 주 처리시스템의 블록다이어그램.

제3도는 그래픽 코프로세서를 수용한 게임 카트리지의 구성과 주 프로세싱 시스템을 수용하고 있는 베이스유닛의 예시적인 기계적 구성을 도시한 사시도.

제4(a)도 및 제4(b)도는 본 예시적 실시예에 따른 그래픽 코프로세서의 블록다이어그램.

제5도는 그래픽 코프로세서의 동작을 개시시키는 주 프로세싱시스템에 의해 수행될 동작시퀀스를 설명하는 플로우챠트.

제6도는 제4(a)도의 산술논리 연산장치의 보다 상세한 블록다이어그램.

제7도는 제4(a)도에 도시된 타입의 예시적인 픽셀 플롯회로의 보다 상세한 블록다이어그램.

제8(a)도는 플롯제어기가 수신한 입력신호와 그 플롯제어기가 출력한 출력 신호를 도시한 블록다이어그램.

제8(b)도는 픽셀 플롯회로의 칼라매트릭스 회로내에 포함된 구성요소들을 나타낸 도면.

제8(c)도는 픽셀 플롯회로와 관련된 타이밍신호, 제어신호 및 데이타신호를 나타낸 도면.

제9도는 제4(a)도의 RAM제어기의 보다 상세한 블록다이어그램.

제9(a)도는 제9도의 RAM제어기와 관련된 타이밍신호, 제어신호 및 데이타신호를 도시한 도면.

제10도는 제9도의 중재논리회로를 설명하는 회로도.

제11도는 본 발명의 그래픽 코프로세서의 예시적 실시예에서의 재동기회로의 다이어그램.

제12도는 제11도의 재동기회로와 관련된 타이밍신호를 나타낸 도면.

제13도는 본 발명의 그래픽 코프로세서에서의 ROM제어기의 보다 상세한 블록다이어그램.

제14도는 본 발명의 예시적실시예에 따른 그래픽 코프로세서에서의 캐시제 어기의 블록다이어그램.

제15(a)도는 본 발명에서의 그래픽 코프로세서의 명령디코딩 관련회로를 나타내는 블록 다이어그램.

제15(b)도는 제15(a)도의 룩어헤드 논리회로의 동작을 설명하는 예시적인 타이밍신호를 나타내는 도면.

제16도 및 제17도는 본 발명의 예시적 실시예에 따른 그래픽 코프로세서의 레지스터 제어논리회로를 나타내는 블록다이어그램.

제18도는 다각형 형성작업을 실행시 그래픽 코프로세서의 동작시퀀스를 설명하는 예시적인 플로우차트.

제19도, 제20도 및 제21도는 본 발명의 예시적 실시예에 따른 스케일링 및 회전특성을 설명하기 위하여 발생된, 다각형 기초대상체의 예시적인 디스플레이를 나타낸 도면이다.

* 도면의 주요부분에 대한 부호의 설명

1, 18 : 커넥터 2 : 마리오칩
3 : 인증프로세서 4 : 외부수정진동자
6, 8 : RAM 10 : ROM
20 : 제어테크 22 : 주 CPU
24 : PPU 26 : APU
28, 32 : W-RAM 30 : V-RAM
34 : RF 변조기 50 : ALU
52 : 플롯하드웨어 60 : 명령디코더
64 : 승산기 66 : 부분곱저항기
68 : 캐시제어기 72 : 캐시태그레지스터
74 : 직접데이타레치 76 : 레지스터블록
78 : 레지스터 제어논리 94 : 캐시 RAM
98 : 게트 B레지스터 152 : 16비트 가감산기
164 : 1비트 6 ×1 멀티플렉서 166 : CPU플래그회로
210 : 비트펜딩회로 212 : 3 ×8 디멀티플렉서
206 : 8 ×8 칼라 매트릭스회로 208 : 비트플레인 카운터
216 : 플롯어드레스레지스터 218 : 어드레스비교기
300 : 16비트 데이타레지스터 304 : MUX
306 : 16비트 MUX 레지스터 314 : 데이타뱅크레지스터
316 : 스크린 뱅크레지스터

[발명의 상세한 설명]

본 발명은 프로그램가능프로세서가 내장된 특수한 외부메모리 유닛을 포함하고 있는 정보처리장치에 관한 것이다.

특히, 본 발명은 일부는 주처리 시스템, 예를들면 비디오게임시스템에 의해 실행되고, 일부는 그 주처리시스템의 고속그래픽 처리능력을 확장시키도록 설계된 프로그램가능 마이크로프로세서에 의해 실행되는 프로그램을 저장하기 위한 프로그램메모리를 갖는 이동식 외부메모리유닛에 관한 것이다.

본 출원은 명칭 "비디오게임 시스템 등에서 사용하기 위한 픽셀/문자변환 하드웨어를 갖는 프로그램 가능프로세서" 로 산(San)등에 의해 동시에 출원된 건과, 명칭 "비디오게임 시스템 등에서 사용하기 위한 확장메모리 제어회로를 갖는 그래픽프로세서" 로 샌등에 의해 출원된 건과 관련된다.

비디오게임 제어데크내에 내장되어 있는 8비트 마이크로프로세서 및 관련 디스플레이 처리서브시스템을 구비하고 있는 종래기술의 비디오게임기는 8×8비트 매트릭스형태로 게임카트리지내에 문자를 사전에 기억시킴으로써, 그리고 사전에 기억된 그 문자의 다양한 프로그램가능조합을 사용하여 스크린 디스플레이를 생성함으로써 그래픽을 형성해 왔다.

일반적으로, 그러한 종래기술의 비디오게임시스템에서 사용자에 의해 조정되는 다수의 "이동체" 나 "스프라이트"뿐만 아니라 전체적인 디스플레이 후면을 이동시킬 수 있었다.

그러나, 종래 기술의 그러한 비디오게임시스템은 회전되어야하고 그리고 프레임에 대해 철회되어야만 하는 다각형의 조합으로 구성된 이동체를 포함하는 비디오게임을 실제로 수행할 수 있는 능력이 없다.

이 시스템에 포함된 종래 8비트 프로세서 및 관련 디스플레이 처리회로는 3차원이며 다각형 기초회전체를 회전시키거나 이 회전체를 3차원의 특수효과를 발생시키도록 스케일링하는데 필요한 계산을 수행할 수 없다.

본 발명자들은 정교한 그래픽은 픽셀바이픽셀에 기초한 스크린을 갱신하고 실시간에 기초하여 복잡한 수치계산을 수행하는 것이 요구됨을 인식하여 왔다.

문자에 기초를 둔 종래기술의 비디오게임기는 그러한 기능을 수행할 수 없다.

또한 종래의 8비트 비디오게임기는 픽셀바이픽셀에 기초한 스크린을 빠르게 갱신하는 것이 요구되는 다른 그래픽기술을 효과적으로 수행할 수 없다.

예를들면 그러한 시스템은 보다 정교한 칼라발생과 보다 좋은 그래픽 해상도를 제공한다.

그러한 16비트 비디오게임기는 문자에 기초를 둔 그래픽이나 스프라이프 그래픽이 가능한 다양한 비디오게임을 제공하는 문자에 기초를둔 시스템이다.

또한 이 16비트 비디오게임기는 다수의 칼라후면 플레인의 뒤 또는 앞에 배치된 이동체와 함께 고속으로 그 플레인들을 이동시키는 것이 가능하다.

그러나, 그러한 종래의 16비트 비디오게임기로는 각 프레임동안 변해야만 하는 다각형으로 구성된 정교한 대상체를 디스플레이하는 3차원 특수효과를 갖는 진보된 비디오게임을 실시할 수 없다.

예를들면, 프레임 바이 프레임 기초상에서 확대 및 축소되어야만 하는 다수의 완전회전체나 스프라이트를 필요로하는 게임은 실제로 그러한 종래 기술의 문자기초 16비트 게임기에서는 실현할 수 없다.

본 발명자들은 완전히 회전하고 스케일링되는 다각형 기초대상체를 포함하는 그러한 게임을 효과적으로 실시할 수 있기 위해서는 다각형의 모서리를 그리는 것과 이러한 다각형 기초대상체에 픽셀바이픽셀 기초상에서 적절한 데이타를 채우는 것이 필요함을 인식하였다.

픽셀바이픽셀 기초상에서 행해져야하는 그러한 작업은 상당히 많은 처리 시간을 소비한다.

종래 기술에서, 이동게임 카트리지는 주 마이크로프로세서와 관련된 어드레스라인의 갯수가 허용하는 것보다도 더 큰 프로그램 메모리어드레스 공간을 존재하는 프로세서가 어드레싱하게 함으로써 게임의 정교성을 향상 시키도록 수정되어 왔다.

예를들면, 그러한 종래 8비트시스템은 메모리뱅크스위칭 및 기타 부가적 인 기능을 수행하는 멀티메모리 제어기칩을 포함하는 게임카트리지를 사용해 왔다.

그러나, 상기 메모리뱅크스위칭 관련칩은 비디오게임시스템으로 하여금 상기 성질의 고속그래픽처리를 수행하게 할 수 없다.

여기서 기술된 그래픽 마이크로프로세서와 비디오게임시스템은 우수한 특징들을 많이 가지고 있는데 몇가지를 여기서 약술한다.

본 발명의 특수그래픽프로세서는 주 마이크로프로세서와 플러그 접속된다.

그 그래픽프로세서는 처리속도를 최대로하기 위해 상기 마이크로프로세서와 병렬로 동작한다.

한 예시적실시예에서는 그래픽코프로세서를 내장하고 있는 게임카트리지가 ROM과 RAM을 또한 포함하고 있다.

본 발명의 그래픽코프로세서는 자기와 주 마이크로프로세서간의 메모리 거래를 중재한다.

그리고 그 그래픽 코프로세서는 종래의 비디오게임시스템에서는 불가능했던 프로그램의 고속처리를 가능하게 하기위해 주 마이크로프로세서와 동시에 프로그램을 실행할 수 있다.

또, 그 그래픽코프로세서는 게임카트리지에 포함되어 있는 3버스구조와 함께, 즉 3개의 버스에 연결되어 동작한다.

부언하면, 그 3버스구조는 주 마이크로프로세서와 그래픽코프로세서의 능력, 즉 RAM과 ROM을 효율적으로 사용하는 능력을 최적화함으로써 그 RAM과 ROM을 효율적으로 사용케한다.

사용자가 전적으로 프로그래밍하여 사용할 수 있는 본 발명의 상기 그래픽코프로세서는 고속으로 처리되도록 설계된 특수한 명령세트를 포함하고 있다.

즉, 그 명령세트는 3차원 그래픽과 관련된 산술동작을 효율적으로 실행하도록 설계되었으며, 예를들면 그 명령세트는 주 비디오게임시스템의 문자맵핑 디스플레이에 개개 픽셀을 플롯시키는 기능을 가진 전용하드웨어에 의해 실행되는 특수명령을 포함하고 있다.

명령세트중의 대부분의 명령은 1기계어 싸이클기간내에 실행될 수 있으며, 그리고 프로그램 ROM의 1바이트내에 저장될 수 있다.

그리고 이들 명령들은 특수목적으로 사용되는 프리픽스명령을 사용함으로써 더욱 강력해질 수 있다.

또, 그 명령 세트는 비록 주시스템 이 문자에 기초하고 있더라도 개개 픽셀에 어드레스를 지정하는 것을 가능케 함으로써 프로그래머 관점에서 "가상" 비트맵을 작성하는 특수픽셀 기초명령들을 포함하고 있다.

그래픽프로세서는 주문자기초 16비트머신이 전형적으로 이용하는 형태의 문자데이타로 픽셀데이타를 계속 변경시킨다.

따라서, 예를들면 프로그래머가 픽셀을 플로팅하기 위하여 "PLOT" 명령을 사용할지라도 관련데이타가 읽

혀지면 그 데이타는 16비트 주머신이 사용할 수 있는 문자기초형태로 변경된다.

특수목적용 픽셀플로팅하드웨어는 고속 3차원그래픽이 효율적으로 실행되도록 하기 위하여 이 명령을 사용한다.

예를들면 그러한 시스템은 3차원 공간에서 디스플레이된 대상체의 일부분인 다각형상에 다른 임의 대상체를 효과적으로 맵핑시킬 수 없다.

그래픽능력을 증래 8비트머신 이상으로 향상시키기 위하여 노력한 끝에 보다 강력한 16비트 프로세서를 사용하는 비디오게임시스템이 고안되었다.

그러한 16비트 프로세서는 그 비디오게임시스템에 보다 정교한 그래픽에 필요한 수치계산을 수행하는 메카니즘을 제공한다.

또한 본 발명의 그래픽코프로세서는 프로그램 ROM에 저장되어 있는 프로그램 명령이 그래픽코프로세서에 의해 캐시 RAM으로부터 고속으로 실행되는 것을 가능케하는 특수한 "캐시(CACHE)" 명령과 캐시메모리 메카니즘을 포함하고 있다.

상기 캐시명령은 고속으로 실행될 프로그램의 일부를 나타냄으로써 프로그래 머가 그래픽코프로세서의 내부 캐시 RAM에 기억되어 있는 프로그램의 실행을 개시하는 것을 가능케 한다.

또한 명령세트는 정교한 3차원형 비디오게임을 실시할 때에 요구되는 그래픽 기술을 프로그래밍하는데 도움되도록 고안된 특수목적 레지스터를 포함하고 있다.

그러한 명령들은 다른 레지스터에 저장된 스프라이트데이타를 합병하여 더 욱 효율적으로 디스플레이된 대상체 회전시키거나 텍스쳐맵핑을 가능케하도록 안된 상기 픽셀플롯(PLOT)명령과 머지(MERGE)명형을 포함하고 있다.

특수목적명령은 데이타를 버퍼링하여 본 발명의 주 마이크로프로세서 및 그래픽코프로세서가 데이타를 병렬처리하는 것을 가능케 한다.

예를들면, 특수목적 명령은 데이타처리속도를 증가시킴으로써 게임카트리지 에 사용된 ROM의 비교적 느린 액세스타임을 보상한다.

이때, 그래픽프로세서는 명령, 즉 소정의 범용레지스터(예를들면, 예시적 실시예에서 레지스터 RI4)에 대한 어떤 레퍼런스(reference)가 ROM으로부터 데이타를 페치하는 동작을 자동적으로 개시시키도록하는 명령을 사용한다.

이렇게 ROM을 액세스하는 동작이 수행되는 동안에 다른 코드가 실행된다.

그리고 몇 싸이클 뒤에야 페치된 데이타가 이용가능하다.

그러나, 한편 프로세서는 그 페치된 데이타를 이용할 수 있을때까지 기다릴 필요가 없으며 대신에 초고속으로 실행되는 코드가 기록될 수 있도록 다른 작업을 할수 있다.

본 발명의 그래픽코프로세서는 서브루틴연결을 효율적으로 다루기 위하여 (LINK)명령, 즉 서브루틴이 완료된 후에 실행될 명령의 어드레스를 완료 시점에서 프로그램카운터(RI5)로 로딩시키는 명령을 또한 포함하고 있다.

명령세트는 RAM스토어 백(store-back)명령을 포함하고 있다.

이 명령에 따라서 ROM으로부터 데이타를 읽어내어 이 데이타에 대해서 동작이 수행된 후에 그래픽코프로세서의 RAM제어기는 바로전에 사용된 RAM어드레스에서 갱신된 데이타스토어 백 동작을 개시시킨다.

이 1싸이클 스토어백명령은 데이타블록을 효율적으로 갱신하는데 편리하게 사용될 수 있다.

본 발명의 그래픽코프로세서는 또한 명령, 즉 MSB가 뒤따르는 LSB를 사용하여 RAM에 대해 기록 및 판독을 가능케하는 명령을 포함하고 있다.

이 메카니즘은 어떠한 데이타변환도 하지않고 두 포맷으로 저장된 데이타와 호환성을 제공함으로써 프로그래밍하는데 도움준다.

본 발명의 그래픽코프로세서는 내부프로세서의 상태레지스터를 변경시킴으로써 다수의 플로팅모드로 세팅될 수 있다.

이러한 다수의 모드는 프로그램 가능명암효과를 발생시키는 혼합모드를 포함하고 있는데, 여기에서 각각 다른 픽셀들은 다른 칼라를 포함하고 있다.

그리고, 또다른 모드, 즉 선택가능모드는 두 스프라이트가 하나의 스프라이트가 차지하고 있는 메모리영역에 저장되도록 칼라에 대한 상위니블과 하위 니블중 하나를 선택하는 것을 가능케 한다.

본 발명은 다수의 하드웨어적 특성을 가지고 있다.

예를들면, 그래픽코프로세서는 칩에 내장되어 있는 RAM을 사용하여 확장 픽셀데이타를 버퍼링시키는 특수목적 플로팅회로를 포함하고 있다.

데이타를 버퍼링시키면 외부데이타 RAM과 리드 또는 라이트하는 거래량이 줄어들며, 그리고 디스플레이될 다각형들이 적절한 데이타로 채워지는 속도가 증가하게 된다.

상기한 바와같이, 레지스터(RI4)를 액세스함에 따라 개시되는 리드버퍼링 특성 이외에, 본 발명의 그래픽코프로세서는 마리오칩의 중앙처리장치가 가능한한 빠르게 다른 명령을 실행할 수 있도록 게임카트리지 RAM에 라이트될 데이타가 버퍼링되는 라이트버퍼링 특성을 가지고 있다.

또한, 본 발명의 그래픽코프로세서는 그래픽프로세서와 주 처리시스템이 쉽게 접근가능한 16개의 레지스터(R0-RI5)를 포함하고 있다.

레지스터(R0)는 명령에서 일부러 지정되지 않아도 되는 디폴트레지스터이며 동시에 누산기로 사용되는 레지스터이다.

레지스터(RI5)는 프로그램 카운터로 사용되는 레지스터이다.

레지스터(RI4)는 ROM으로부터 데이타를 패치하는 동작을 자동적으로 개시시키는 레지스터이다.

특수프리픽스명령은 소스레지스터 및 행선지 레지스터를 설정하는데 사용되는 명령이다.

본 발명의 그래픽코프로세서는 주코프로세서와 상호작용하며 그 결과 주프로세서가 그래픽코프로세서의 레지스터를 쉽게 액세스할 수 있다.

그래픽코프로세서와 관련된 3버스구조는 수준높은 병렬성(parallelism)을 제공해준다.

이들 버스들은 구조적으로 떨어져있어 동시에 사용될 수 있다.

이들 각 버스는 어드레스버스, 데이타버스 및 제어버스이다.

주프로세서버스는 그래픽코프로세서가 필요로하는 다양한 신호를 제공하기 위한 어드레스라인, 데이타라인 및 제어라인을 포함하고 있다.

이 버스구조를 사용하는 본 발명의 그래픽코프로세서는 프로그램 ROM이나 RAM 또는 자체내에 캐시 RAM에 저장되어 있는 프로그램을 실행할 수 있다.

그래픽코프로세서는 여러가지 중재모드를 사용하고 있는 주 마이크로 프로세서와 인터페이스 한다.

즉, 이 때 논리 "1" 을 소정의 그래픽코프로세서의 상태 레지스터 위치에 로딩시킴으로써 중재모드가 주프로세서에 의해 세팅되어 주프로세서가 게임카트리지 ROM 및 RAM를 액세하는 동작을 포기했음을 나타내게 된다.

본 발명자들은 주프로세서가 상태레지스터를 적절히 세팅시킴으로써 ROM 및 RAM을 액세스하는 동작을 포기한 상황에서조차도, 인터럽트가 발생되어 주프로세서가 이 인터럽트를 처리하기 위한 루틴의 어드레스를 페치하고자 RAM액세스동작을 개시함을 알았다.

이러한 상황하에서, 주 마이크로프로세서가 자기자신의 내부 작업RAM을 액세스할 수 있도록 그래픽프로세서는 주 마이크로프로세서에서 작업 RAM어드레스를 프로그램 ROM어드레스 대신에 제공한다.

이 기술은 프로그램 RAM에 저장되어 있는 프로그램 이 그래픽코프로세서에 의해 실행되고 있는 중에 주프로세서가 프로그램 ROM을 어드레싱하지 못하게 한다.

주프로세서가 카트리지 RAM을 액세스할 필요가 있을때는 그래픽 코프로세서의 상태레지스터가 세팅되써 이 그래픽코프로세서가 RAM을 액세스 할 수 없기때문에 주프로세서는 RAM으로부터 필요로하는 정보를 액세스하고 후에 그래픽코프로세서로 하여금 RAM을 액세스할 수 있는 상태로 전환시킬 수 있다.

그러나 코프로세서는 처리속도를 향상시키기 위해 가능한한 카트리지 ROM 및 RAM을 사용하는 것이 바람직하다.

본 발명의 그래픽코프로세서는 문자데이타 RAM의 픽셀정보를 디스플레이용 주프로세서 비디오 RAM으로 효율적으로 전송시킬 수 있도록 설계되었다.

그러나, 비디오 RAM은 그래픽코프로세서가 어떤 카트리지 버스를 통해 직접 액세스할 수 없게 되어있다.

그러한 정보전송은 주프로세서의 직접 메모리 액세스(DHA)회로를 사용하여 행해져야 한다.

본 발명의 그래픽코프로세서는 주정보처리 시스템으로부터 여러가지 클록신호를 수신한다.

그래픽코프로세서내에서의 타이밍은 이들 수신된 클럭신호중 하나의 신호로 구동된다.

본 발명의 옵션특성으로서, 그래픽코프로세서내의 회로는 파워온리세트 직후에 구조설정 입력라인으로 사용되는 출력 어드레스라인을 통해 수신된 신호의 상태에 따라 그 그래픽코프로세서가 장래의 변경에 대처하도록 재구성 될 수 있도록 되어있다.

이들 어드레스라인에 연결된 옵션설정 저항기의 값은 그래픽코프로세서가 판독하게 된다.

이들 신호는 예를들면 그래픽프로세서와 함께 사용되고 있는 RAM칩이 타입, 즉 스태틱 RAM이냐 다이나믹 RAM이냐를 설정하는데 사용되고 있다.

본 발명의 제반특성들은 첨부도면과 함께 고찰하면 더 잘 이해될 것이다.

본 예시적실시예에 따르면, 본 발명의 그래픽코프로세서는 닌텐도 오브 아메리카사가 판매하고 있는 모델명 "수퍼 닌텐도 오락시스템(Super NES)" 인 16비트 비디오게임시스템과 대화한다.

그 수퍼 닌텐도 오락시스템은 명칭 "비디오처리장치" 로 1991년 4월 10일자로 출원된 미국 출원 No. 07/651,265와 명칭 "직접메모리 액세스장치 및 이 장치에 사용되는 외부기억 디바이스" 로 1991년 8월 26일자로 출원된 미국출원 No. 07/749.530에 일부 기재되어 있다.

이들 출원건은 여기에서 특별히 참조한다.

여기서 주지해야할 사항은 본 발명은 수퍼 NES관련 출원에 한정되는 것이 아니고 다른 비디오게임시스템은 물론 비디오게임시스템이 아닌 정보처리장치에도 사용될 수 있다는 것이다.

참조하기 쉽도록 본 예시적실시예에 따른 그래픽프로세서를 이하 "마리오 칩(Mario chip)" 이라 칭한다.

본 예시적실시예에서 그 마리오칩은 비디오게임카트리지 내에 내장된다.

본 발명에서는 마리오칩이 프로그램 메모리와 주처리장치에 접속되는 한 마리오칩은 프로그램메모리와 동일한 카트리지케이스내에 내장되지 않아도 된다.

제1도에 본 발명의 예시적실시예에 따른 예시적 비디오게임 카트리지와 외부메모리시스템이 도시되어 있다.

그 비디오게임카트리지는 제1도에 도시된 모든 부품이 부착될 PCB(도시 안됨)를 포함하고 있다.

그 비디오게임카트리지는 PCB의 입력단에 설치되어 수퍼 NES 메인콘트롤 데크와 신호를 주고 받기위한 커넥터 전극어레이(1)를 포함하고 있다.

커넥터 전극어레이(1)는 수퍼 NES메인 콘트롤데크에 설치된 상대커넥터와 접속된다.

본 예시적실시예에 따라서, 비디오게임카트리지에 내장된 마리오칩(그래픽 코프로세서)(2)은 100~128개의 핀을 갖는 IC칩이다.

그 마리오칩(2)은 주처리시스템(즉, Super NES)으로부터 많은 제어신호, 어드레스신호 및 데이타신호를 수신한다.

예를들면, 마리오칩(2)은 주처리시스템으로부터 핀(P112)을 통해 21MHZ의 클럭입력신호와 핀(P117)을 통해 21MHZ(또는 다른 소정의 주파수)의 시스템클록 입력신호를 수신한다.

이 시스템클럭 입력신호는 예를들면 마리오칩에 주 CPU메모리액세스에 대한 메모리타이밍정보를 제공하는데 사용되며 그리고 마리오칩내에서의 타이밍동작에 클럭신호를 제공하는데 사용된다.

마리오칩(2)은 또한 선택적인 외부클럭입력핀(P110)을 가지고 있는데 이 핀(P110)에는 예를들면 주시스템으로부터 수신된 21MHZ보다 더 높은 주파수클럭레이트를 갖는 신호를 발생시켜 마리오 CPU를 구동시키기 위한 외부수정진동자(4)가 연결된다.

주처리시스템(즉, 수퍼 NES CPU/영상처리장치(PPU))의 주 CPU어드레스입력핀(HA)은 핀(P37-P62)을 통해 마리오칩(2)에 연결된다.

유사하게, 주시스템의 데이타입력핀(HD)은 주 CPU 데이타버스로부터 핀(P65-72)을 경유하여 마리오칩(2)에 연결되다.

마리오칩(2)은 부가적으로 주 CPU로부터 핀(P119)을 통해 메모리 리프레쉬신호(RFSH)를, 핀(P118)을 통해 리세트신호를, 그리고 핀(P104)(P105)을 통해 리드 및 라이트 제어신호를 각각 수신한다.

마리오칩은 인터럽트 요구신호(IRQ)를 발생시켜 핀(P120)을 경유하여 수퍼 NES에 그 IRQ를 제공한다.

마리오칩은 그 에 다른 제어신호도 수신하는데 예를들면 주프로그램 ROM(10)을 액세스하는 동작을 개시시키기 ROMSEL신호를 수신한다.

게다가, 게임카트리지는 입력(I)라인, 출력(0)라인 및 리세트(R)라인상에서 수퍼 NES인증(authentication)프로세서와 데이타를 교환하는 기능을 가진 인증프로세서(3)를 포함하고 있다.

게임카트리지를 인증하는데 사용되는 인증프로세서(3)와 보안시스템은 여기서 참조하고 있는 미합중국 특허 No. 4,799,635호에 게재된 타입의 것이다.

마리오칩의 RAM어드레스핀(P74-P91)은 RAM어드레스버스(RAMA)를 통해 RAM(6)(8)에 연결되고, 마리오칩의 RAM데 이 타핀(P93-P100)은 RAM데이타버스 (RAM D)를 통해 RAM(6)(8)에 연결된다.

이들 RAM(6)(8)은 각각 마리오칩의 핀(P90)(P91)을 통해 제공된 행어드레스 스트로브신호와 열어드레스 스트로브신호(RAS)(CAS)를 사용하여 부분적으로 제어되는 다이나믹메모리 디바이스이다.

그리고 그 다이나믹 RAM대신에 하나이상의 스태틱 RAM이 사용될 수 있으며 이때에는 핀(P90)(P91)이 행어드레스 및 열어드레스 스트로브신호없이 어드레스 신호를 그들 각각의 스태틱 RAM에 제공하는데 사용될 수 있다.

쓰기가능 제어신호(WE)는 핀(P107)을 통해 RAM(6)(8)에 제공된다.

읽기제어신호 및 쓰기제어신호(R)(W)는 주 CPU에서 발생되어 핀(P104) (P105)을 통해 마리오칩에 제공된다.

마리오칩은 이들 읽기라인 및 쓰기라인을 감시함으로써 수퍼 NES CPU가 수행하려하는 메모리액세스동작의 종류를 알아낼 수 있다.

비슷하게, 주시스템의 사실상 모든 어드레스라인 및 제어라인은 주 CPU가 수행하려하는 동작의 트랙을 보유하고 있는 마리오칩에 의해 감시된다.

마리오칩이 수신한 ROM어드레스신호와 RAM어드레스신호는 감시되고 난후에 적절한 메모리 디바이스에 기억되게 된다.

이때 ROM어드레스는 ROM어드레스버스와 핀(P2-P26)을 통해 프로그램 ROM(10)에 기억되고, RAM어드레스는 핀(P74-P91)을 통해 RAM(6)(8)에 기억된다.

주 CPU의 ROM데이타입력과 RAM데이타입력은 각각 ROM데이타버스 및 핀(P28-P35)과, 핀(P93-P100)를 통해 ROM(10)에 기억된다.

한가지 주지해야할 사항은 마리오칩은 지금까지 기술한 ROM과 RAM뿐만 아니라 그외 다양한 메모리디바이스와도 연결된다는 것이다.

예를든다면, 마리오칩은 CD ROM을 사용하는 비디오게임시스템에 연결된다.

예를들면, 제1도에서 ROM(10)을 사용하는 대신에 CD ROM(도시 안됨)이 문자데이타, 프로그램명령, 비디오데이타, 그래픽데이타 및 음성데이타를 저장하는데 사용될 수 있다.

종래의 CD플레이어(도시안됨)는 데이타버스(P28-P35)상으로 데이타 및 명령을 액세스할 목적으로 어드레스버스(P2-P26)상으로 메모리어드레스 신호를 수신하기 위하여 마리오칩(2)에 연결되었다.

CD플레이어나 CD ROM 기억시스템의 특수한 구조나 동작에 대해서는 이 분야에서 통상의 지식을 가진자에게 잘 알려져 있다.

CD ROM 기억장치의 한 가지 이점은 정보의 바이트 당 필요한 기억장치의 비용이 상당히 감소된다는 것이다.

즉, 데이타는 반도체 ROM에서보다도 100% 내지 1000% 정도로 저렴하게 저장될 수 있다.

그러나, 불행하게도 CB ROM의 메모리액세스시간 및 메모리리드시간이 반도체 ROM보다 훨씬 느리다.

마리오칩은 적어도 3개의 버스상에 있는 정보가 병렬로 처리하는 것을 가능케하는 3버스구조를 이용하고 있다.

즉, 제1도에 도시된 게임카트리지에서 마리오칩(2)은 ROM데이타라인, ROM어드레스라인 및 ROM제어라인을 포함하는 ROM 버스와 : RAM데이타라인, RAM어드레스라인 및 RAM제어라인을 포함하는 RAM 버스와 : 주데이타라인, 주어드레스라인 및 주제어라인을 포함하는 주프로세서라인과 연결되어 있다.

마리오칩구조는 파이프라이동작이 처리능력을 최적화하는 것을 가능케 한다.

즉, 마리오칩은 다른 데이타를 처리하면서 ROM으로부터 임의 데이타바이트를 읽어낼 수 있음과 동시에, 3차원 관련 그래픽을 매우 효과적으로 수행하기 위해 또 다른 데이타를 RAM에 기록함으로써 처리능력의 최적화

를 도모하고 있다.

후술하겠지만, 마리오칩(2)은 내부적으로 16비트구조를 사용하면서도 8비트 ROM(10)칩 및 8비트 RAM(6)(8)칩과 인터페이스 할수 있도록 설계되어 있다.

그리고, 마리오칩의 모든 내부데이타버스와 내부레지스터는 16비트이다.

또한, ROM(10)으로부터 데이타를 읽어내는 동작과 RAM(6)(8)에 데이타를 기록하는 동작은 버퍼링 되어있지만 이 버퍼링으로 인해 프로그램 실행 속도는 늦어지지 않는다.

비슷하게 마리오칩(2)은 CD ROM으로부터 명령 및 그래픽데이타를 액세스하여 RAM(6)(8)에 기록하는데 이는 주프로세서, 가령 수퍼 NES영상처리장치(PPU)의 비디오 RAM으로 다음 DMA를 전송하기 위함이다.

이 분야에서 통상의 지식을 가진자라면 마리오칩(2)은 RAM기억동작 및 RAM액세스동작을 바이패스시켜 처리함으로써 CD ROM으로부터 PPU의 비디오 RAM으로 데이타를 직접 전송하는 것을 조정하도록 프로그램되어질 수 있다.

CD ROM의 액세스시간이 길다는 사실에도 불구하고 CD ROM를 그래픽에 실제 응용할 수 있는 것을 마리오칩(2)의 처리속도가 초고속이기 때문이다.

비디오데이타 및 오디오데이타는 CD ROM에 저장되기전에 종래의 데이타 압축 기술로서 압축된다.

데이타압축기술과 데이타비압축(decomlpression)기술은 이 분야에서 통상의 기술을 가진자에게 잘 알려져 있다.

마리오칩(2)은 압축된 데이타를 CD ROM으로부터 액세스한 후에 종래 그래픽 프로세서로 달성될 수 있는 시간보다 훨씬 더 짧은 시간내에 종래 데이타비압축알고리즘로써 데이타를 비압축한다.

마리오칩(2)은 21MHZ클록으로 작동되기 때문에 데이타를 RAM(6)(8)으로 전송하는데 요구되는 소정의 시간내에 비압축을 완료한다.

따라서, 많은 양의 비디오데이타와 오디오데이타가 일반적인 CD ROM액세스 시간내에 압축된 형태로 액세스된다.

그러나, 이러한 비교적 긴 액세스시간으로 인해 생기는 영향은 최소화될 수 있는데 이것은 데이타 바이트당 실제 액세스시간이 마리오칩(2)에 의해 데이타가 비 압축된후에 상당히 감소되었기 때문이다.

마리오칩(2)이 비압축을 수행하고 있더라도 주 그래픽프로세서, 즉 수퍼 NES PPU는 다른 처리작업을 자유로이 수행할 수 있다.

물론, 속도가 문제가 되지않는 특정한 응용에서는 마리오칩(2)은 압축되지 않은 형태로 데이타를 CD ROM으로부터 액세스할 수 있다.

또한 스태틱 RAM이 사용되면 게임카트리지는 백업배터리를 포함하게 된다.

백업배터리(12)는 데이타절감특성을 제공하는데 전력손실이 생긴 경우에 스태틱 RAM용 백업 전압(RS RAM) 및 스택틱 RAM칩선택신호(RAMCS)를 제공하기 위하여 저항기(R)를 통해 종래 백업배터리회로(14)에 연결된다.

부가적으로, RAM어드레스버스에 옵션설정저항기(16)가 연결된다.

보통 동작에서, 마리오칩 어드레스라인은 RAM(6)(8)에 출력을 제공한다.

그러나, 리세트동작이나 파워-온 동작시에 그 어드레스라인은 소정의 전압치인 VCC에 연결되느냐 아니면 그라운드(OV)에 연결되느냐에 따라 하이신호 또는 로신호를 발생시키는 입력라인으로서 사용된다.

이런식으로 "1" 또는 "0" 이 내부마리오칩 레지스터로 읽혀지게 된다.

리세트후에, 마리오칩은 이들 저항기의 설정에 따라 가령 승수클럭비, 즉 마리오칩이 연결되는 RAM액세스시간을 프로그램 실행동안에 결정하며, 그 클록비는 마리오칩등의 다른 동작과 더불어 사용된다.

이들 옵션설정 레지스터를 사용함으로써 가령, 마리오칩을 어떠한 설계상의 수정없이 다수의 다른 형태의 메모리디바이스와 사용가능하다.

예를들어 다이나믹 RAM설정이 검출되면 적절한 횟수의 리프레쉬신호가 가해진다.

게다가, 옵션설정은 가령 프로세서 곱셈회로의 동작속도를 제어하는데 사용될 수 있고, 그리고 다른 명령이 어떤 특정곱셈 명령실행보다 더 빠른 레이트로 그래픽프로세서에 의해 실행되도록하는 것을 가능하게 하는데 사용될 수 있다.

따라서 지연된 곱셈실행을 개시함으로써 잔여명령이 다른 가능한 클럭레이트보다 더 빠른 클럭레이트로 실행될 수 있다(예를들면, 프로세서는 30MHZ로 클럭되고, 반면에 옵션설정은 곱셈명령이 15MHZ에서 효과적으로 실행되도록 한다).

제2도는 제1도에 나타낸 전형적인 게임 카트리지가 연결되도록 설계된 전형적인 주 비디오게임시스템의 블록다이어그램이다.

예를들면 제2도는 닌텐도 오브 아메리카사가 현재 판매하고 있는 수퍼 NES를 나타낸다.

그러나 본 발명은 제2도에 도시된 것과같은 블록다이어그램을 갖는 수퍼 NES관련 응용품이나 시스템에 제한되지 않는다.

수퍼 NES는 예를들면 65816호환성 마이크로프로세서인 16비트 주 CPU(22)를 수퍼 NES의 제어테크(20)내에 가지고 있다.

주 CPU(22)는 예를들면 128K바이트의 작업 RAM(W-RAM)(32)에 연결되어 있다.

또, 주 CPU(22)는 예를들면 32K워드의 비디오 RAM(V-RAM)(30)에 연결된 영상처리장치(PPU) (24)와 연결된다.

주 CPU(22)는 수직 또는 수평블랭킹구간 동안 PPU(24)를 경유하여 비디오 RAM(30)에 접근하는 통로를 가지고 있다.

따라서, 주 CPU(22)는 PPU(24)가 비디오 RAM을 액세스하고 있을때 이외의 시간대에서 PPU(24)를 통해 비디오 RAM(30)을 액세스할 수 있을 뿐이다.

PPU(24)는 비디오 RAM(30)으로부터 사용자의 텔레비젼(36)상에 비디오 화상을 발생시킨다.

또한, CPU(22)는 작업 RAM(W-RAM)(28)에 연결된 음성처리장치(APU)(26)와 연결된다.

시판되고 있는 사운드칩을 구비하고 있는 APU(26)는 게임카트리지상의 ROM(10)에 저장된 비디오게임프로그램의 실행과 관련하여 음성을 발생시킨다.

CPU(22)는 단지 APU(26)를 통해 작업 RAM(28)을 액세스할 수 있다.

PPU(24)와 APU(26)는 RF변조기(34)를 통해 사용자의 가정텔레비젼(36)에 연결되어 있다.

수퍼 NES내에 있는 비디오 RAM(30)은 카트리지내의 프로그램 ROM(10)에 기억되어 있는 문자데이타로 적재되어야만 한다(프로그램 ROM(10)은 게임 프로그램뿐만 아니라 게임이 플레이되는 동안 사용될 문자데이타도 저장하고 있다).

어떤 이동체, 예를들면 스프라이트정보, 또는 디스플레이될 배경정보는 사용전에 비디오 RAM(30)에 기억되어 있어야 한다.

프로그램 ROM(10)은 제1도의 인쇄회로기판 모서리커넥터(1)와 접속되는 제2도의 결합커넥터(18)를 통해 주 어드레스버스 및 주데이타버스 상에서 CPU(22)에의 해 액세스된다.

PPU(24)는 공유주 CPU데이타버스 및 공유주 CPU어드레스버스와 커넥터(23)를 거쳐 게임카트리지와 접속되는데 그 이유는 이 게임카트리지에 제공할 PPU데이타신호 및 제어신호를 전송하기 위한 통로를 제공하기 위함이다.

APU(26)는 공유주 CPU버스 및 오디오버스(27)를 거쳐 게임카트리지와 접속 된다.

CPU(22)의 어드레스공간은 프로그램 ROM(10)의 위치가 위치 0에서부터 시작 되도록 맵핑되고, 전형적으로 32K바이트 세그먼트로 나뉘어진다.

그 프로그램 ROM(10)은 대략 CPU어드레스공간의 1/2을 사용한다.

일반적으로 CPU어드레스공간의 각 32K바이트 세그먼트에서 최고위치는 작업 RAM(32)과 여러 레지스터의 어드레스지정을 위해 이동된다.

프로그램 ROM(10)의 기억용량은 대개 4메가바이트이다.

수퍼 NES에서 사용되는 CPU(22)는 프로그램 ROM(10)전체의 어드레스를 지정할 수 있다.

한편, 마리오칩(2)은 단지 16비트 프로그램 카운터만을 가지고 있기 때문에 프로그램용(10)내의 32K바이트 뱅크와 32K바이트뱅크간을 선택하기 위한 뱅크레지스터를 포함하고 있다.

본 예시적인 실시예에서 마리오칩은 수퍼 NES메모리맵과 대응하는 전체 24비트 어드레스공간을 가지고 있다.

이것은 위치 $00 : 8000에서 시작하는 위치에 ROM(10)을 포함하며, 카트리지 상의 RAM(6)(8)은 위치 $70 : 0000에서 시작한다.

카트리지상의 ROM(10)과 RAM(6)(8)은 별도의 버스를 이용하고 있으므로 마리오칩에 의해 병렬로 액세스될 수 있다.

또한 RAM(6)(8)은 ROM보다도 더 빠른 속도로 액세스될 수 있는데 마리오칩은 이 성능상의 장점을 활용하도록 설계된다.

마리오칩(2)은 수퍼 NES내에 있는 어떠한 메모리, 즉 작업 RAM(32)이나 비디오 RAM(30)에 접근할 수 있는 어떤 통로로 가지고 있지 않다.

마리오칩(2)이 데이타를 처리하기 위해서는 또는 비트맵을 작성하기 위해서는 데이타가 RAM(6)(8)내에 저장되어 있어야만 한다.

따라서, NES CPU프로그램과 마리오칩 프로그램이 공유하고 있는 어떤 변수들이 RAM(6)(8)내에 존재하여야 한다.

마리오칩프로그램이 필요로하는 사전에 기억된 어떤 데이타는 ROM(10)에 존재할 수 있고, 반면 어떤 변수는 RAM(6)(8)내에 존재할 것이다.

수퍼 NES프로그램에서만 필요한 전용변수는 RAM(6)(8)내에 존재할 필요가 없다.

사실상, 이 RAM(6)(8)은 메모리공간에 의해 지나치게 비싸므로 반드시 필요한 변수만이 저장되어야할 것이다.

반드시 필요한 변수가 아닌 변수는 수퍼 NES의 내부에 RAM(32)에 기억되어야만한다.

마리오칩(2)이 작성한 비트맵은 마리오카트리지의 RAM(6)(8)내에 저장되고 그리고 각 비트프레임이 전부완성되었을때 수퍼 NES의 제어하에서 비디오 RAM(30)을 DMA전송될 것이다.

수퍼 NES의 CPU(22)는 마치 마리오칩이 존재하지 않는것처럼 수퍼 NES의 제어데크(2D)내에 있는 모든 내부 RAM에 접근하는 통로를 가지고 있다.

마리오칩은 이들 RAM에 접근하는 통로를 가지고 있지 않으므로 마리오 ROM/RAM과 수퍼 NES RAM간에 전송될 모든 데이타는 CPU(22)자체에 의해 해결, 제공되어야 한다.

데이타는 DMA전송을 통해 프로그래밍되거나 블록이동된 CPU(22)을 경유하여 전송될 수 있다.

마리오카트리지의 ROM(10)과 RAM(6)(8)은 모든 게임프로그램상에서 평소와 같이 맵핑된다.

CPU(22)는 카트리지의 ROM(10)과 RAM(6)(8)에 일시적으로 접근할 수 있는 통로제공하는 제어장치를 가지고 있다.

파워(전원)가 상승하거나 리세트시에 마리오칩은 턴오프되고, CPU(22)는 ROM(10)과 RAM(6)(8)으로의 통로를 점유하게 된다.

마리오칩이 프로그램을 실행하기 위해서는 CPU(22)의 프로그램이 ROM(10)이나 RAM(6)(8)에의 바람직하게는 이들 모두에의 접근을 포기하여 마리오칩이 주어진 작업을 완료할때까지 기다리거나, 또는 CPU(22)가 일부코드를 작업 RAM(32)에 카피하여 이곳(작업 RAM(32))에서 실행하는 것이다.

마리오칩은 수퍼 NES의 CPU(22)가 프로그래밍하고 읽어낼 수 있는 것이 가능한 많은 레지스터를 가지고 있다.

이들 레지스터들은 위치 $00 : 3000에서 시작하는 CPU(22)의 메모리맵으로 맵핑된다.

제2도에 도시된 바와같이 수퍼 NES는 다양한 제어신호를 송수신한다.

수퍼 NES의 CPU(22)는 ROM(10)을 액세스하고자할 때 제어신호(ROMSEL)를 발생시킨다.

수퍼 NES(22)는 메모리리프레쉬를 개시하고자할때 리프레쉬신호(RFSH)를 발생시킨다.

마리오칩은 동작을 완료하면 수퍼 NES의 CPU에 연결된 인터럽트 리퀘스트 라인상으로 인터럽트신호(IRQ)를 전송한다.

부가적으로 CPU(22)는 리드신호와 라이트신호를 발생시킨다.

시스템타이밍신호는 제어테크(20)내에 있는 타이밍 체인회로(21)에서 발생된다.

파워-온/리세트신호도 제어테크(20)내에서 발생되어 게임카트리지에 제공된다.

또한, 수퍼 NES는 입력(I)도체, 출력(O)도체 및 리세트(R)도체상의 데이타를 상기 미합중국 특허 No.4,799,635호에 따른 게임카트리지상의 중재프로세서(3)와 교환하는 기능을 갖는 인증프로세서(25)를 포함하고 있다.

미합중국 특허 No.4,799,635호에 게재된 인증프로세서(25)는 인증이 설정될 때까지 CPU(22)를 리세트상태로 유지시킨다.

제2도에 블록으로 나타낸 수퍼 NES비디오게임기는 여기에서 일반적으로만 기술되어진다.

예를들면, PPU(24)를 포함하는 수퍼 NES에 관한 상세한 설명은 1991년 4월 10일자로 "비디오처리장치"의 명칭을 가지고 출원된 미합중국 출원 No.07/651,265호에 게재되어 있으며, 참고로 여기에서 인용한다.

가령, 수퍼 NES와 게임카트리지간에 정보가 어떻게 전송되는가등과 같은 더 자세한 설명은 "영상처리시스템의 직접 메모리 액세스장치 및 그 장치에 사용된 외부기억장치"의 명칭으로 1991년 8월 26일에 출원된 미합중국 특허출원 No.07/749,530호와, 발명의 명칭 "모자이크 영상디스플레이장치와 이에 사용된 외부기억장치"로 1991년 11월 19일에 출원된 미합중국 출원 No.07/793,735호에 게재되어있고, 여기에서 이들을 참조한다.

본 발명자들은 일부응용에서 실제로 가능한 것보다 더 많은 정보가 그러한 주프로세서 DMA회로를 사용하는 수직블랭킹 동안에 전송되어야할 필요가 있음을 같이 인식하였다.

따라서, 수직블랭킹시간을 확장함으로써 영상사이즈가 축소되는 결과를 가져올지라도 그 수직블랭킹시간을 확장하는 것이 바람직하다.

이로써 처리속도와 영상갱신비에 있어서 상당한 이익이 실현된다.

제3도는 제1도에 도시된 마리오칩과 다른 카트리지 구성요소를 수용하기 위한 게임카트리지케이스(19)의 하나의 실시예를 나타낸 사시도이다.

또한, 제3도는 제2도에 도시된 수퍼 NES비디오 게임하드웨어를 수용하기 위한 비디오게임제어테크(20)에 대한 예시적인 외부하우징의 사시도이다.

이러한 비디오게임 제어테크(20)와 이동가능한 게임카트리지(19)의 기계적인 설계에 대하여는 여기에서 참조될 명칭 "TV게임기"로 1991년 8월 23일자로 출원된 미합중국 출원 No.07/748,938호의 제2도 내지 제9도에 도시되어 있다.

제4(a)도 및 제4(b)도는 제1도에 도시된 마리오칩(2)의 룰럭다이어그램이 다.

먼저, 제4(a)도 및 제4(b)도에 도시된 여러 가지 버스에 대해 설명한다.

명령버스(INSTR)는 명령코드를 여러 마리오칩 구성요소에 제공하는 8비트버스이다.

X버스, Y버스 및 Z버스는 16비트 데이타버스이다.

HA버스는 본 실시예에서 수퍼 NES어드레스버스에 연결되는 24비트 주 시스템 어드레스 버스이다.

HD버스는 수퍼 NES의 데이타버스에 연결되는 8비트 주데이타버스이다.

PC버스는 마리오칩의 프로그램카운터(즉, 일반적인 레지스터블록(76)의 레지스터(RI5))의 출력을 여러시스템 구성요소에 제공하는 16비트버스이다.

ROM A버스는 20비트 ROM어드레스버스이다.

ROM D버스는 8비트 ROM데이타버스이다.

RAM A버스는 1비트 RAM어드레스버스이다.

RAM D_IN버스는 8비트 RAM리드데이타버스이고, RAH D_OUT버스는 8비트 RAM 라이트데이타버스이다.

마리오칩과 수퍼 NES는 마리오칩과 수퍼 NES간에 데이타를 전송하기 위한 메인메카니즘의 역할을 하는 카트리지의 RAM(6)(8)을 공유한다.

수퍼 NES는 어드레스버스(HA)와 데이타버스(HD)를 통해 마리오칩을 액세스 한다.

수퍼 NES는 수퍼 NES어드레스버스(HA)를 통해 마리오칩의 레지스터(76)를 액세스한다.

수퍼 NES는 마리오칩(2)을 통해 카트리지프로그램 ROM(10)과 RAM(6)(8)을 액세스한다.

ROM제어기(104)와 RAM제어기(88)는 각각 ROM과 RAM의 액세스를 개시하기 위하여 수퍼 NES에 의해 발생된 메모리액세스 관련신호를 수신한다.

예로써, 마리오칩(2)은 수퍼 NES가 RAM의 어드레스지정을 시도하고 있음을 알리기 위하여

RAM선택신호(RAMCS)를 사용한다.

제4(a)도 및 제4(b)도에 도시된 X,Y,Z버스는 마리오칩의 내부데이타버스이다.

X버스와 Y버스는 소스데이타버스이고, Z데이타어스는 행선지 버스이다.

이들 버스들은 16비트의 명령데이타를 전송한다.

마리오칩(2)은 명령어를 실행하면서 X버스와 Y버스상에 명령실행에 사용되는 데이타의 소스를, Z버스상에 행선지데이타를 각각 싣는다.

가령, 두 레지스터에 기억되어 있는 내용을 가산하여 그결과를 어떤 제3의 레지스터에 기억시키라는 명령을 실행할때에, 산술논리연산장치(ALU)(50)는 X버스와 Y버스를 통해 두 소스레지스터의 내용을 수신하여 연산한후, 그 연산결과를 Z버스에 싣는다(차례로 이 Z버스는 레지스터(76)에 연결되어 있다).

마리오칩(2)내의 명령디코딩회로(60)로 명령의 옵코드를 디코딩하여 얻는 제어신호는 ADD연산을 개시시킬 목적으로 ALU(50)에 제공된다.

제1도에서 기술한 바와같이 마리오칩은 병렬로 신호를 통신할 수 있는 ROM버스, RAM버스 및 수퍼 NES주버스에 연결된다.

마리오칩(2)은 주시스템이 수행하고 있는 동작을 알아내기 위하여 주수퍼 NES버스를 통해 전송된 제어신호, 어드레스신호 및 데이타신호를 감시한다.

카트리지 ROM버스와 카트리지 RAM버스는 어떤 주어진 시간에 수행되고 있는 수퍼 NES동작에 따라서 병렬로 액세스된다.

종래의 수퍼 NES게임 카트리지에서, 주 CPU어드레스라인과 데이타라인은 RAM과 ROM에 직접 연결되기 때문에 RAM과 ROM은 병렬로 액세스되지 않는다.

본 발명의 한측면에 따라서, 마리오칩(2)은 제1도에 도시된 ROM버스와 RAM버스를 수퍼 NES버스로부터 구조적으로 분리시킨다.

마리오칩(2)은 수퍼 NES버스상으로 전송된 신호를 감시하여, 시분할되지 않은 두 별도의 ROM버스와 RAM버스를 통해 어떤 신호가 ROM칩과 RAM칩에 연결되어야 하는지를 결정한다.

ROM버스와 RAM버스를 분리시킴으로써 마리오칩(2)은 ROM를 리드하는 것과 RAM에 라이트하는 것을 동시에 할수 있다.

이로써, 마리오칩은 RAM를 액세스하기에 앞서 ROM의 액세스가 완료될때까지 기다릴 필요없이, RAM액세스시간보다 상당히 느린 액세스시간을 갖는 값싼 ROM칩으로도 효과적으로 작동될 수 있다.

제4(a)도에서, 마리오칩(2)은 상기한 바와같이 전적으로 프로그램가능한 프로세서이며, ALU(50)를 포함하고 있다.

ALU(50)는 승산기(64)에 의한 곱셈 연산과 플롯하드웨어(52)에 의한 특정 픽셀플로팅연산을 제외하고 마리오칩내에서 수행되는 모든 산술연산을 실행한다.

ALU(50)는 명령어 디코더로부터 적절한 제어신호를 수신하여 가산연산, 감산연산, 배타적-OR연산 및 시프트연산등을 수행한다.

제4(a)도에 도시된 바와같이, ALU(50)는 X버스와 Y버스로부터 처리될 정보를 수신하고, 명령 디코더(60)의 제어신호에 의해 개시되는 연산을 수행하고, 그리고이 연산결과를 Z버스에 제공한다.

이제 ALU가 제6도에 참조하면서 보다 상세하게 기술된다.

마리오칩(2)은 3차원의 특수한 효과 및 다른 그래픽동작이 효과적으로 수행될 수 있도록 하기위한 특수목적의 하드웨어를 또한 포함하고 있어, 이들 특징을 활용하는 비디오게임이 실제적으로 실현될 수 있다.

이 때, 마리오칩(2)은 실시간에서 픽셀좌표 어드레싱을 문자맵 어드레싱으로 변환시키는데 도움이되는 플롯하드웨어(52)를 포함하고 있다.

마리오칩은 디스플레이 스크린상에 있는 각 픽셀의 위치를 설정하는 X좌표와 Y좌표를 지정함으로써 편리하게 프로그램되어진다.

따라서, 그래픽 동작은 프로그래머가 지정한 픽셀에 기초하여 수행되며, 플롯하드웨어회로(52)는 픽셀사양을 정확히 포맷된 문자데이타로 변환시킨다.

그리고나서, 그 문자데이타는 디스플레이되기 위해 제2도의 수퍼 NES 비디오 RAM(30)의 원하는 위치로 맵핑된다.

이 때, 마리오칩프로그래머는 문자맵핑시 수퍼 NES비디오 RAM(30)을 비트맵 으로 간주할뿐이다.

플롯하드웨어(52)는 디스플레이 스크린상의 X및 Y좌표와 특정픽셀의 소정의 칼라를 프로그램가능하게 선택하는 것을 가능케하며, 대응픽셀을 플롯팅하여 X 및 Y좌표가 수퍼 NES비디오 RAM(30)를 구동시키는데 사용되는 형태의 문자 정의에 대응하는 주소로 변환시키는 여러 플로팅 관련명령에 응답한다.

플롯하드웨어(52)는 카트리지 RAM에 라이트하기에 앞서 가능한한 많은 픽셀데이타를 버퍼링시킴으로써 RAM데이타의 상호작용을 최소화시키는 기능을 갖는 데 이타래치와 연결되어 있다.

문자정의데이타는 X좌표데이타와 Y좌표데이타가 변환되어 플롯하드웨어 (52)에 버퍼링되고난 후에 카트리지 RAM으로 전송된다.

플롯하드웨어(52)는 PLOT X레지스터(56)와 PLOT Y레지스터(58)를 통해 X좌표데이 타 Y좌표데이타를 각각 수신한다.

본 실시예에서, PLOT X레지스터와 PLOT Y레지스터는 제4(a)도에 도시된 바와같이 별개의 레지스터가 아니고 마리오칩의 범용레지스터(예를들면, 제4(b)도의 레지스터출력(76)내의 레지스터(R1)(R2))이다.

또, 프롯드하드 웨어(52)는 칼라레지스터(54)를 통해 픽셀칼라정보를 수신한다. 후술되겠지만, 디스플레이될 각 픽셀의 칼라은 8 ×8레지스터 매트릭에 저장되는데, 이때 각 픽셀칼라정보는 1컬럼(column)의 매트릭스를 차지한다.

플롯하드웨어(52)는 X, Y는 칼라입력에 관련된 문자어드레스와 문자데이타를 처리하여 문자 RAM(6)(8)에 입력시킨다.

문자어드레스는 출력라인(53)을 통해 RAM제어기(88)와 RAM어드레스버스 (RAM A)에 차례로 전송된다.

문자데이타는 출력라인(55), 멀티플렉서(93) 및 RAM 데이타버스 RAM D-OUT를 통해 문자 RAM에 제공된다.

플롯하드웨어(52)는 수퍼 NES문자포맷과 호환성을 유지하면서 프로그래머에게 "가상" 비트맵디스플레이 시스템을 제공하기 위하여 문자내의 픽셀이 개별적으로 주소지정되는 것을 가능케한다.

그 "가상" 비트맵은 카트리지 RAM에 기억되어 있으며, 각 프레임의 디스플레이를 완료하기 위해 예를들면 전술한 출원 No.07/749,530의 DMA 회로를 사용하여 수퍼 NES비디오 RAM(30)에 전송된다.

플롯하드웨어(52)는 개별픽셀을 고속으로 제어하는 것이 가능함으로 회전체와 스케일링체를 포함한 특정 3차원 그래픽효과가 실제로 실현될 수 있다.

픽셀을 문자형태로 변환시키는 것 때문에, 플롯하드웨어(52)는 카트리지 RAM(6)(8)으로부터 RAM-in 데이타래치(82)와 입력 라인(83)을 통해 현재의 픽셀(X)(Y)근처에 있는 다른 픽셀과 관련된 정보를 또한 수신한다.

RAM에 라이트하는 횟수는 RAM(6)(8)으로부터 검색되어 RAM데이타래치에 일시적으로 저장된 이전의 픽셀 데이타를 사용함으로써 최소화될 수 있다.

또한, 제4(a)도에 도시된 RAM데이타 래치(80,84,86)는 플롯하드웨어(52)에 그러한 데이타를 제공하기 위하여 카트리지 RAM내의 많은 비트플레인(bit plane)에 저장되어 있는 픽셀에 대해 수신된 칼라데이타를 버퍼링하는데 사용된다.

RAM데이타래치(80)는 수퍼 NES가 RAM데이타래치(80)의 내용을 읽어낼수 있도록 수퍼 NES데이타버스에 연결되어 있다.

RAM 데이타래치(80, 82, 84, 86)는 RAM 제어기(88)로 제어된다.

RAM 데이타래치(84)(86)는 RAM(6)(8)의 데이타를 수신하여 이 데이타를 레지스터블록(76)내의 소정의 레지스터에 로딩시키기 위하여 행선지 Z버스에 제공한다.

부가적으로 RAM어드레스를 버퍼링시키는 래치(90)가 RAM제어기(88)에 연결된다.

RAM 제어기(88)는 RAM A버스를 통해 RAM(6)(8)의 주소를 지정하기 위하여 래치(90)내에 저장되어 있는 어드레스를 사용한다.

또, 수퍼 NES는 어드레스버스(HA)를 통해 RAM 제어기(88)를 액세스한다.

플롯하드웨어(52)는 레지스터(R1)의 내용에 의해 설정된 수평위치에 대한 픽셀칼라정보와 레지스터(R2)의 내용에 의해 설정된 수직위치에 대한 픽셀 픽셀정보를 리드하라는 리드픽셀(READ PIXEL)명령에 응답하여, 그 결과를 행선지 Z버스와 출력라인(87)을 통해 레지스터블록(76)내의 소정의 레지스터에 저장시킨다.

이 플롯하드웨어(52)는 뒤에 제7도, 제8(a)도 및 제8(b)도를 참조하면서 보다 상세히 기술된다.

파이프라인 버퍼레지스터(62)와 ALU 제어기 명령디코더(60)는 명령버스(INSTR)에 연결되어 명령버스에 실려있는 명령에 응답하여 동작을 개시시키기 위한 제어신호(CTL)(이 신호는 마리오칩 전체적으로 이용됨)를 발생시킨다.

마리오칩(2)은 현재의 명령을 실행하면서 다음에 실행될 명령을 페치하는 기능을 갖는 파이프라인된 마이크로프로세서이다.

파이프라인 레지스터(62)는 가능하면 1싸이클내에 명령이 실행될 수 있도록 명령을 저장하고 있다.

명령어버스상의 명령은 레지스터, 가령 제4(b)도의 레지스터블럭(76)내의 레지스터(R15)에 저장되어 있는 프로그램카운터의 내용에 의해 주소가 지정된다.

마리오칩(2)에 의해 실행될 명령은 제1도의 프로그램 ROM(10) 또는 마리오 칩의 내부캐시 RAM(94) 또는 카트리지 RHA(6)(8)에서 얻는다.

프로그램 명령이 ROM(10)에 저장되어 있다면 ROM제어기(104)(제4(b)도)는 이 ROM(10)에서 명령을 페치하여 마리오칩의 명령 버스(INSTR)상에 싣는다.

프로그램 명령이 캐시 RAM(94)에 저장되어 있다면, 그 명령은 캐시 RAM(94)으로부터 캐시 RAM출력버스(95)를 통해 명령버스상에 직접 실리게 된다.

주 CPU, 즉 수퍼 NES는 마리오칩 프로그램명령을 위해 프로그램 ROM(10)의 일부를 할애하도록 프로그램되어진다.

수퍼 NES프로그램은 마리오칩에게 소정의 기능을 수행하도록 지시하고나서, 마리오칩 프르그램코드를 액세스하기 위해 마리오칩에 ROM(10)에 저장되어 있는 어드레스를 제공한다.

파이프라인 레지스터(62)는 룩어헤드(lookahead)관련처리를 가능케하는 프로그램 실행중에 일어날 일을 디코더가 예측할 수 있게하는 명령관련 정보를 명령디코더(60)에 제공하기 위하여 실행될 명령 앞의 명령 1바이트를 페치한다.

블록(60)내의 디코딩 및 제어회로는 ALU(50), 플롯하드웨어(52), 캐시제어(68)등에서 실행중인 명령코드에 의해 지시된 동작을 수행할 것을 명령하는 제어신호를 발생시킨다.

또, 마리오칩은 ALU(50)와는 별도로 고속형 병렬승산기(64)를 포함하고 있 다.

그 승산기(64)는 소정의 명령에 응답하여 X소스버스와 Y소스버스로부터 각각 수신된 2개의 8비트수를 곱셈연산하여, 그 16비트 결과를 행선지 Z버스에 싣는다.

이 곱셈연산은 가능한한 1싸이클내에 수행되게 된다.

승산기(64)에로의 숫자입력은 부호가 붙거나 붙지 않는다.

또한, 승산기(64)는 긴 곱셈연산을 수행할 수 있으므로 2개의 16비트수가 곱해져 32비트의 수를 낳는다.

그리고 승산기(64)는 곱셈연산중에 발생된 부분곱을 저장하기 위하여 연결된 부분곱레지스터(66)를 포함하고 있다.

승산기(64)는 곱셈 연산코드가 디코드될 때 명령 디코더(60)의 제어신호에 의해 인에이블 된다.

승산기(64)는 16비트워드의 곱셈을 포함하는 긴 곱셈명령을 최소한 4클럭 싸이클내에 실행한다.

긴 곱셈명령은 다음의 형태를 가지고 있다 :

R4(로워드), DREG(하이 워드) = Sreg * R6.

이 명령은 소스레지스터(Sreg)와 레지스터(R6)의 내용을 곱하여 그 32비트 결과를 레지스터(R4/DREG)(로/하이)에 기억시킴으로써 실행된다.

이 곱셈은 부호가 붙게되고, 32비트 결과치에 0및 부호플래그가 설정된다.

이 연산은 다음의 6단계로 행해진다.

단계 1 : 부호없는 곱셈 R4 [0...15] = SREG [0...7] * R6 [0...]

단계 2 : X에 부호가 붙음.

R4 [0...15] = R4 [0...15] + 256 * SREG [9...15] * R6 [0...7].

곱셈결과의 상위 8비트는 무시되지만 가산중의 캐리는 보존된다.

단계 3 : X에 부호가 붙음.

R5 [0...15] = (Y + [R6 [8...15] * SREG [0-7]) ÷ 256 : 부호확장.

단계 4 : X에 부호가 붙고 Y에 부호가 붙지 않음.

R4 [0...15] = R4 [0...15] + 256 * SREG [0...7] * R6 [8...15].

곱셈 치의 상위 8비트는 무시되지만 가산으로부터의 캐리는 보존된다.

단계 5 : Y에 부호가 붙음.

R5 [0...15] = R5 [0...15] + (Y+ SREG [0...7] * R6 [8...15].

÷ 256 : 부호확장.

단계 6 : X와 Y에 부호가 붙음.

R5 [0...15] = R5 [0...15] + RY [8...15] * R6 [8...15].

본 실시예의 승산기(64)는 가령 카바노(Cavanaugh)가 지은 맥그르우 힐 (McGraw-Hill)출판사의 1984년판, 디지탈컴퓨터 아리스메틱(Digital Computer Arithmetic)에 게재되어 있는 타입의 승산기이다.

제4(b)도에서 캐시제어기(68)(제14도에서 상세히 설명됨)는 프로그래머가 고속으로 실행되기를 원하는 일부 프로그램을 캐시 RAM(94)에 로딩(loding) 시키는 것을 효율적으로 개시할 수 있게 한다.

그러한 "캐싱(caching)" 은 대개 그래픽처리에서 자주 일어나는 작은 프로그램루프를 실행할시에 이용된다.

마리오칩 명령세트에는 "캐시(cache)" 명령이 포함되어 있다.

캐시명령의 바로뒤의 명령은 캐시 RAM이 꽉 찰때까지 캐시 RAM에 로드(load)된다.

개시명령이 실행되면 현재의 프로그램카운터상태가 캐시베이스 레지스터(70)에 로드된다.

따라서, 캐시베이스레지스터(70)의 내용은 캐싱이 시작되는 시작위치를 나타낸다.

대부분의 명령은 1싸이클내에서 실행 된다.

RAM(10)이나 RAM(6)(8)과 같은 비교적 느린 외부메모리로부터의 명령은 실행되기 전에 페치된다.

이것은 6싸이클정도를 더 필요로한다.

프로그램 실행속도를 증가시키기 위해서는 마리오칩 자체내에 있는 캐시 RAM(94)이 사용되어야 한다.

캐시 RAM(94)은 512바이트 명령캐시 RAM인데 이것은 보통 프로그램의 사이즈보다 비교적 작은 사이즈이므로, 프로그래머는 이 캐시메모리(94)를 어떻게 잘 사용할지를 결정해야만 한다.

512바이트의 캐시사이즈에 맞는 프로그램루프를 전속력으로 페치하고 실험 하는데 1싸이클이면 된다.

그리고, 버스가 분할되었기 때문에 ROM 및 RAM은 내부캐시 RAM(94)에 기억되어 있는 코드가 실행되는 동안도 동시에 액세스될 수 있다.

캐시 RAM(94)은 캐시 RMA(94)에 기억되어 있는 루프를 실행시킴으로써 스프라이트를 회전시키는데 편리하게 사용될 수 있다.

그 루프는 회전 및 스케일링계산을 수행하면서 ROM(10)으로부터 각 픽셀의 칼라를 읽어내는 내용의 루프이며, 대조적으로 RAM(6)(8)에 라이트하고자 할때는 상기 루프로써 하는게 아니라 PLOT명령을 사용하여 라이트동작을 수행한다.

즉, 상기 모든 동작은 병렬로 처리되어, 가장 느린동작으로 인해 더디어진 처리능력을 매우 빠르게 한다.

보통 상기 가장느린 동작은 ROM에 기억되어 있는 데이타를 페칭하는 동작인데, 이유는 마리오칩이 ROM과 RAM에 접근하기 위한 버퍼된 통로를 사용하도록 설 계되었기 때문이다.

비교적 느린 ROM(10)에서의 프로그램실행과 비교하여 볼 때, 이 프로그램은 캐시 RAM(94)에서 약 6배나 더 빠르게 실행되지만, 먼저 그 프로그램이 ROM(10)으로부터 캐시 RAM(94)으로 로드되어야만 한다.

이것은, 즉 ROM(10)에서 캐시 RAM(94)으로의 로딩은 명령을 캐시될 루프의 맨앞에 배치시킴으로써 행해진다.

따라서 이 루프의 캐시명령의 어드레스로 지정된 최초 512바이트만이 개시될 것이다.

그 프로그램은 루프의 최초반복을 위한 코드를 실행하는 동안 ROM(10)으로부터 페치되어 16바이트 청크(Chunk)로 캐시 RAM(94)에 메모리될 것이다.

루프의 계속된 모든 반복은 ROM(10)대신에 캐시 RAM(94)에서 수행된다.

캐시(CACHE)명령은 어떠한 반복적인 프로그램루프의 앞부분에서 자유롭게 사용될 수 있다.

루프의 계속된 반복은 캐시 덕택이다.

만약 프로그램루프가 512바이트보다 더 커서 캐시 RAM(94)의 용량을 초과하더라도 프로그램루프는 정상동작될 수 있지만, 최초 512바이트만이 캐시 RAM(94)에서 실행되며 나머지 바이트는 평소와 같이 ROM(10)에서 실행된다.

이것은 부분적으로 속도를 상승시키지만 이상적인 것은 아니다.

본 실시예에서 캐시 콘트롤러(68)의 구성요소인 캐시 태그비트 레지스터(72)는 캐시 RAM(94)에 로드되어 있는 메모리위치를 식별한다.

캐시 태그비트는 프로그램명령이 프로그램 ROM(10)에서 보다 고속인 캐시 RAM에서 실행가능한지를 마리오칩이 빠르게 결정할 수 있게 한다.

캐시 RAM(94)은 수퍼 NES버스(HA)와 멀티플렉서(96)를 통해 캐시 제어기(68)나 수퍼 NES에 의해 액세스된다.

캐시제어기(68)는 캐시 베이스 레지스터(70)를 로드하기 위해 프로그램 카운터 버스(PC)에 연결되며, 캐시 메모리 어드레스가 범위를 이탈했는지의 여부를 체크하는 동작을 수행한다.

ROM(10)을 병렬로 리드(read)하는 것과 유사하게, 마리오칩은 RAM(6)(8)에 병렬로 라이트(Write)하는 한가지 방법도 제공한다.

마리오레지스터가 RAM(6)(8)에 라이트될때는 언제나, 메모리 트랜잭션은 예를들면 RAM제어기(88)내의 별도 RAM라이트회로를 개시시킬 것이다.

이것은 전형적으로 6싸이클이 소요되지만, 프로그래머가 그때에 또다른 RAM트랜잭션을 수행하지 않는다면 처리기를 지연시키지 않을 것이다.

예를들면 각 기억명령 사이에서 다른 처리를 실시하는 것이 더 빠르다.

이 방식으로 RAM라이트회로는 작업을 할 시간을 갖게 된다.

예를들면(아래 명령세트의 명령을 사용한다) :

```
FROM    R8      ;  R8을 (R13)에 기억시킴
SM     (R13)
SM     (R14)    ;  R0을 (R14)에 기억시킴
TO      R1
FROM    R2
ADD     R3      ;  r1 = r2 + r3를 수행
TO      R4
FROM    R4
ADD     R6      ;  r4 = r5 + r6를 수행
```

2개의 기억 명령은 서로 너무 가깝게 배치되어 있음에 주의하자.

제2기억명령은 RAM버스가 제1기억명령을 완료하는데 분주하기 때문에 6싸이클이상을 필요로 한다.

더 빠르게 실행될 코드를 라이트하는 더 좋은 방법은 2개의 기억명령을 다른 유용한 코드와 간격을 두어 배치하는것이다.

예를 들면 :

```
FROM    R8      ;  R8을 (R13)에 기억시킴
SM     (R13)
TO      R1
FROM    R2
ADD     R3      ;  r2 = r2 + r3을 수행
TO      R4
FROM    R5
ADD     R6      ;  r4 = r5 + r6을 수행
SM     (R14)    ;  R0을 (R14)에 기억시킴
```

이 방법으로, 제 1기억명령이 RAM에 라이트하는 것과 동시에 명령이 몇 개 더 병렬로 실행될 수 있다.

후술된 명령어세트는 레지스터를 최종적으로 사용된 RAM 어드레스에 다시 기록시킬 때 사용하는 고속명령을 포함하고 있다.

이 고속명령은 RAM에 기억되어 있는 값을 로딩하고 어떤 처리를 실시하고 나서 다시 고속으로 그 값을 기억

시킴으로써 데이타의 벌크처리를 가능케한다.

제4(b)도에서 직접 데이타래치(74)는 명령버스(INSTR)에 연결된다.

이 직접 데이타래치(74)는 명령자체가 데이타소스를 제공하는 것을 가능케 하며 따라서 어떤 소스레지스터 명령에 의해 지정될 필요가 없다.

직접 데이타래치(74)의 출력은 행선지Z버스에 연결되어 있고, 차례로 레지스터블록(76)의 레지스터중 소정의 레지스터에 연결된다.

명령어 디코딩회로(60)는 "직접" 데이타명령을 디코딩하여 레지스터동작에 대한 적절한 전달동작을 개시시킨다.

제4(b)도에 도시된 게트(GET) 레지스터(98)는 전술한 지연, 버퍼된 리드 동작에 사용된다.

이때, 광범위하게 사용되는 비교적 느린 액세스타임을 갖는 ROM들을 사용하면, 대개 종래기술의 프로세서는 ROM을 실행시킬 때마다 페치가 완료될때까지 기다려야만 한다.

후술될 지연, 버퍼된 페치메카니즘을 사용함으로써 데이타를 페치하고 있는 중에도 다른 동작이 수행될 수 있다.

이 메카니즘에 따라서, 레지스터블록(76)내의 레지스터(RI4)가 어떤 방법으로든 액세스되거나 수정되면, ROM 또는 RAM의 페치가 RI4의 대응으로 식별된 어드레스에서 자동적으로 개시된다.

제4(b)도에 나타낸 바와같이, 레지스터(RI4)는 ROM제어기(104)에 연결된다.

레지스터(RI4)의 내용이 어떤 방법으로든 수정되면, ROM제어기(104)는 ROM액세스를 개시하는 작동을 한다.

ROM을 액세스한 결과는 ROM데이타버스(RGHD)에 연결된 멀티플렉서(102)를 통해 게트B 레지스터(98)로 로드된다.

아래에 식별된 명령은 게트 B레지스터(98)에서 버퍼된 정보를 액세스하는 것을 가능케 한다.

이 정보는 멀티플렉서를 통해 행선지 Z버스상에 로드되고나서 레지스터블록(76)내의 하나의 레지스터로 로드된다.

이 방법으로, ROM에서 데이타를 페치하는데는 소정의 처리싸이클의 걸리더 라도 피치동작을 개시될 수 있으며, 그리고 마리오칩은 다른 동작을 수행하지 않으면서 기다리는게 아니고 상기 데이타페치동작이 개시된 후에 예를들면 비관련코드를 실행할 수 있다.

또, 게트 B레지스터(98)는 제4(b)도의 멀티플렉서(102)를 경유하여 R(6)(8)으로부터 검색된 정보를 저장하는데 이용된다.

레지스터블록(76)내에는 16개의 16비트 레지스터(R0-RI5)가 내장되어 있다.

레지스터(R0-R13)는 이들 일부레지스터가 후술된 특수목적으로 자주 사용되더라도 특수목적레지스터가 아니고 범용레지스터이다.

상술한 바와같이, 레지스터(R14)는 메모리를 판독하기 위한 포인터로서 사용되며, 그 레지스터(RI4)가 변경되면 ROM(또는 RAM)에 대한 리드싸이클이 개시된다.

레지스터(RI4)로 판독된(리드된)바이트는 나중에 게트 L명령이나 게트 H명령 에 의해 액세스될 수 있도록 임시버퍼(게트 B레지스터(98))내에 저장된다.

레지스터(BI5)는 프로그램카운터이며 각 명령의 실행이 개시될때 페치될 다음 명령을 가리킨다.

레지스터(RO)는 범용레지스터이며 전형적으로 누산기의 기능을 갖는다.

또한, 그 레지스터(RO)는 대부분의 1싸이클명령을 위한 디폴트소스 및 행선지레지스터이다.

예를들어, 레지스터(RG)의 내용과 레지스터(R4)의 내용을 함께 가산하려면 레지스터(B4)를 분명하게 지정하기만 하면 된다.

레지스터(R11,R12,R13)는 루프명령실행시 특별히 사용되는 레지스터들이다.

레지스터(RI3)는 루프의 맨위에서 실행될 명령의 어드레스를 저장하며, 레지스터(RI2)는 루프가 반복실행될 횟수를 저장한다.

레지스터(R12)의 내용이 0이 아니면 R13의 내용에 의해 지정된 어드레스에 있는 명령은 프로그램카운터(R15)에 로드되어 실행된다.

레지스터(R11)는 루프가 완료되고 난후 복귀할 주소를 저장한다.

레지스터 제어논리회로(78)는 레지스터블럭(76)에 연결되어 범용레지스터(R0-R15)에 대한 액세스를 제어한다.

명령어 디코드논리회로(60)는 실행될 특정 명령의 형식에 따라서 하나이상의 레지스터(R0-R15)를 지정한다.

레지스터 제어논리회로(78)는 실행될 다음명령이 이용하기를 원하는 레지스터를 지정한다.

레지스터 제어 논리회로(78)는 적절한 레지스터의 출력을 X버스와 Y버스에 제공한다.

제4(b)도에 나타낸 바와같이 적절한 레지스터(R0-R1)는 레지스터 제어논리회로(78)의 제어하에서 Z버스로부터 정보를 수신한다.

ROM제어기(104)는 수퍼 NES어드레스버스(HA) 또는 마리오칩으로부터 어드레스를 받는 즉시 그 어드레스를 액세스한다.

ROM제어기(104)가 제13도에 상세하게 도시되어 있다.

ROM(10)으로부터 액세스된 정보를 고속명령실행을 위해 캐시 RAM(94)으로 로드된다.

ROM제어기(104)와 RAM제어기(108)는 수퍼 NES액세스시도와 마리오칩 액세스 시도간을 중재하는 버스중

재유닛을 가지고 있다.

후술되겠지만, 마리오칩은 수퍼 NES CPU에 의해 액세스가능한 상태레지스터를 예를들면 레지스터블록(76) 또는 RAM(6)(8)내에 또한 가지고 있다.

이 상태레지스터는 0플래그, 캐리플래그, 부호플래그, 오버플로우플래그, "고(GO)" 플래그와 같은 상태조건을 식별하기 위한 플래그(여기서, 1은 마리오 칩이 동작하고 있음을 나타내고, 0은 마리오칩이 정지하고 있음을 나타낸다)와 ; 레지스터(RI4)가 액세스되고 있음을 나타내는 ROM바이트페치인 진행플래그와 ; ALT 1 플래그, ALT 2플래그, 직접바이트로 플래그 및 직접바이트 하이플래그와, 소스 및 행선지 레지스터가 "위드(WITH)" 프리픽스명령에 의해 세트되었음을 알리는 플래그와, 인터럽트플래그를 포함하는 여러가지 모드지시플래그를 저장하고 있다.

초당 여러 번의 작업을 수행하기 위하여 마리오칩의 온/오프를 전환시키는 수퍼 NES는 제4(a)/4(b)도에 블럭다이어그램으로 나타낸 마리오칩을 사용하고 있다.

초기에, 수퍼 NES가 턴온되면 ROM(10)에 저장된 게임프로그램이 부팅된다.

게임프로그램이 수퍼 NES 및 마리오칩프로세서로 실행되기 전에 먼저 게임 카트리지가 인증(authentication)되어야 한다는 것에 주의하자.

예로써, 상기 인증은 처음에 수퍼 NES CPU를 리세트시킨 상태에서 미합중국 특허 No. 4,799,635호에 게재된 기술에 따른 게임카트리지 및 수퍼 NES의 메인 제어테크에 연결된 인증프로세서로 인증프로그램을 실행함으로써 이루어진다.

마리오칩은 초기에 스위치오프된 상태에 있게 된다.

이 때, 수퍼 NES는 게임카트리지 프로그램 ROM과 게임카트리지 RAM을 자유로이 액세스할 수 있다.

수퍼 NES는 그래픽동작이나 수학적계산을 수행하기 위해 마리오칩 처리능력을 이용할 필요가 있을때 마리오칩이 처리기를 원하는 적절한 데이타를 카트리지 RAM(또는 소정의 마리오레지스터)에 저장하고, 마리오칩 프로그램카운터에 실행될 마리오프로그램의 어드레스를 르드시킨다.

마리오칩으로 처리될 데이타는 회전되고 확대 또는 축소되어야할 대상물의 소정의 X, Y좌표 데이타이다.

마리오칩은 다수의 스프라이트나 이동체 후면 및 전면을 조작하는 내용의 알고리즘을 수행하는 프로그램을 실행할 수 있다.

마리오칩의 속도증가 하드웨어 및 소프트웨어를 사용하면 그러한 동작을 고속으로 수행할 수 있다.

스프라이트를 처리하는데 마리오칩을 사용하면 전반적인 비디오게임시스템의 능력을 크게 증대시킬 수 있다.

예를들면, 수퍼 NES는 프레임당 128개의 스프라이트를 디스플레이하도록 제한되어 있다.

그러나, 수퍼마리오칩의 사용으로 사실상 수백개의 스프라이트가 디스플레이 되며 예를들면 회전된다.

마리오칩이 수퍼 NES가 요구하는 기능을 완료하고나면 스톱(STOP)명령이 실행되며, 지금 마리오칩이 동작을 완료하고 다음 작업을 수행할 준비가 되었음을 지시하는 인터럽트신호가 발생되어 수퍼 NES에 전송된다.

마리오칩은 고속곱셈과 같은 간단한 작업을 수행하는데 사용되거나 스프라이트로 가득찬 스크린을 그리는데 사용될 수 있다.

어떤 경우에서는 수퍼 NES는 마리오칩이 RAM버스 또는 ROM버스를 사용하고 있는중에 이들 버스를 사용치 않는다면 마리오칩과 병렬로 처리동작을 자유로이 수행할 수 있다.

수퍼 NES가 게임카트리지상에서 RAM버스와 ROM버스의 제어신호를 마리오 칩에 제공할지라도 수퍼 NES는 제2도의 작업 RAM(32)에서 프로그램을 실행시킬 수 있음에 주의하자.

따라서 전반적인 시스템의 처리능력은 마리오칩이 동시에 어떤 프로그램을 처리하고 있는 동안, 실행될 수퍼 NES프로그램을 프로그램 ROM으로부터 작업 ROM으로 카피함으로써 증대된다.

제5도에 마리오칩으로 하여금 원하는 어드레스에 있는 ROM의 코드를 페치, 실행시키게하는, 주 CPU(예를들면, 수퍼 NES CPU)로 실행될 "런 마리오(RUN MARIO)" 프로그램에 의해 수행되는 동작의 시퀀스를 나타낸 플로우차트가 도시되어 있다.

제5도에 나타낸 루틴은 프로그램 ROH(10)으로부터 제2도에 도시된 작업 RAM(32)으로 카피된 후에 수퍼 NES CPU에 의해 실행된다.

이 루틴은 마리오칩이 어떤 동작을 수행하고자 할때마다 주 CPU에 의해 실행된다.

블록 125에 도시된 바와같이 런 마리오(RUH NAR10)주 CPU루틴이 처음 실행될때 수퍼 NES레지스터를 보호하는등의 초기화동작이 수행된다.

초기화단계 동안에 이 런 마리오 주 CPU루턴은 프로그램 ROM(10)으로부터 주 CPU의 작업 RAM(32)으로 카피된다.

블록 127에 나타낸 바와같이 실행될 마리오 프로그램코드를 저장하는 ROM(10)코드뱅크가 마리오칩 레지스터로 로드된다.

게다가, 코드뱅크내의 실제주소는 블록 129에 나타낸 바와같이 마리오칩 스크린베이스 레지스터 내에 저장된다.

그후, 블록 131에 나타낸 바와같이 I/O 입력/출력모드는 4나 16 또는 256칼라 모드중 어느 칼라모드가 사용될 것인가를 식별함으로써 마리오칩에 설정된다.

이들 모드는 주 CPU가 동작하는 칼라모드와 일치한다.

부가적으로 디스플레이될 문자의 갯수에 의해 스크린의 높이를 설정하는 모드가 세트된다.

또, 마리오칩에 ROM버스 및 RAM버스의 제어를 제공하는 모드비트가 세트된다.

ROM버스 및 RAM버스의 제어는 마리오칩이 ROM버스 또는 RAM버스 또는 ROM 버스와 ROM버스모두로의

통로를 가지고 있는 모드에 세트되도록 별도로 선택 가능하다.

따라서, "마리오 오너(Mario owner)" 모드가 ROM 및 RAM에 대해 설정된다면 주 CPU는 ROM 또는 RAM에 대해 리드/라이트할 수 없다.

마리오칩이 프로그램 ROM버스를 사용하고 있는데도 주 CPU가 프로그램 ROM을 액세스하려 한다면 마리오칩이 모조어드레스(dummy address)를 수퍼 NES에게 되돌려주는 메카니즘이 제공된다.

그러한 어드레스로 분기하는 것은 마리오칩이 더 이상 카트리지 ROM버스에의 액세스를 요구하지 않을때까지 수퍼 NES를 점유된채로 유지되게 한다.

블록 133에 나타낸 바와같이 마리오칩은 마리오칩 프로그램카운터가 마리오 루틴이 실행해야만하는 최초명령을 저장한 어드레스로 로드된 후에야 동작을 개시한다.

그리고나서, 주 CPU는 마리오칩으로부터 인터럽트신호가 오기를 기다린다(블록 135).

수퍼 NES는 인터럽트신호가 수신되면 현재 마리오칩이 동작을 완료하여 정지상태에 있음을 알게된다(블럭 137).

그러나 주 CPU가 그러한 어떤 인터럽트신호도 수신받지 못하면 계속 인터럽트신호가 오기를 기다린다(블록 135).

이 시간기간동안에 수퍼 NES는 제2도의 작업 RAM(32)에서 프로그램코드를 마리오칩 동작과 병렬로 실행한다.

그리고나서, 수퍼 NES는 마리오칩이 동작중에 있음을 지시하는 마리오칩의 "고(G0)" 플래그가 세트되었는지 결정하기 위해 상태 레지스터(가령, 마리오칩 레지스터블럭(76)내에 있음)를 체크한다(블록 137).

부가적으로 인터럽트플래그가 마리오칩이 주 CPU가 수신한 인터럽트 신호의 소스임을 지시하기 위하여 마리오칩의 상태 레지스터에 세트된다.

따라서, 주 CPU(135)가 인터럽트신호를 수신한후에 마리오칩에 그 인터럽트의 소스인지를 결정하기 위해 적절한 마리오상태 레지스터가 테스트 된다.

RAM 및 ROM에 대한 마리오 오너 모드비트는 마리오칩이 정지하면 클리어되므로 수퍼 NES가 ROM과 RAM에 대한 액세스를 독점하게 된다.

수퍼 NES는 루틴 141를 빠져나와 런 마리오루틴에 들어가기전에 실행하고 있었던 프로그램으로 복귀한다.

CPU(22)의 게임프로그램은 마리오칩을 ROM마리오 오너모드로 설정할때 자진해서 ROM을 액세스하던 동작을 정지하여야 한다.

CPU(22)가 어떤 이유로해서 ROM을 액세스하고자 할때는 ROM마리오 오너모드를 간단히 던 오프하면 된다.

마리오칩은 다음에 ROM을 액세스할 필요가 있을때 ROM마리오 오너모드가 다시 주어질때까지 자동적으로 지속될 것이다.

마리오칩이 ROM에 대해 마리오 오너모드에 설정되어 있으면 CPU(22)게임 프로그램은 그 ROM으로부터 아무것도 리드(read)할려고 해서는 안된다.

예를들면 수직블랭킹으로 인해 인터럽트가 발생되면, 즉 인터럽트가 NMI를 야기시키면 CPU(22)는 자동적으로 ROM으로부터 인터럽트벡터를 페치하려 할 것 이다.

그러나 이것은 CPU(22)가 마리오칩에게 ROM을 액세스하지 않겠다고 확실히 지시하고서도 이를 어기고 인터럽트를 발생시켜 ROM을 액세스하려 하기 때문에 바람직하지 않다.

이 상황에서, 마리오 오너모드가 설정되어 있음에도 불구하고 CPU(22)가 ROM을 액세스하려하는 것은 마리오칩으로하여금 그것이 인터럽트 벡터요구라고 오해를 불러 일으킬 것이다.

마리오칩은 ROM마리오 오너 모드상태에서 인터럽트벡터를 페치하는 중에 그 인터럽트벡터를 수퍼 NES의 내부에 있는 작업 RAM(32)의 스택영역의 바닥에 재배치시킬 것이다.

예를들면, 인터럽트벡터가 $00 : FFFC라면 그 인터럽트벡터는 위치 $00 : 01OC로의 점프를 일으킬 것이다.

유사하게 $00 : ffex로부터의 모든 인터럽트벡터는 CPU(22)를 $00 : 010X에서 그들의 대응하는 위치로 점프시킬 것이다.

이 기술은 CPU(22)로하여금 ROM(10)을 액세스하지 말고 대신에 수퍼 NES의 RAM(32)을 액세스하게 한다.

한가지 주지해야할 사항은 RAM기초 인터럽트벡터는 인터럽트처리기로의 점프나 분기를 포함해야만 하는데, 즉 실제코드는 간단한 벡터 어드레스가 아닌곳에 있어야 한다는 것이다.

마리오칩이 ROM마리오 오너모드상태에 있지 않으면 보통의 ROM인터럽트벡터 가 사용되며, 그래서 RAM기초 인터럽트벡터와 동일장소로 가게하기 위하여 이들 위 치에서 지적된 동일 어드레스를 유지하는 것이 바람직하다.

이제 명령세트에 대해서 기술한다.

마리오칩 명령은 고속그래픽스 및 다른 처리알고리즘을 프로그래밍하는데효율적인 수단을 제공한다.

일부 명령들에 대한 간단한 설명이 여러명령에 의해 사용되는 일부레지스터에 대한 설명과 함께 설명된다.

또한 명령세트의 명령에 대한 상세한 리스트도 포함하고 있다.

명령들은 8비트로서 대개 1클록싸이클내에 실행된다.

그러나, 그 명령들은 8비트 프리픽스(prefix)명령들에 의해 수정될 수 있다.

마리오칩 명령 세트는 프로그래머가 어떤 명령앞에 행선지 및 두 소스레지스터를 지정하는 것을 가능케하는 특정 레지스터 오우버 라이드 시스템(override system)을 포함하고 있다.

그러한 "프리픽스" 오우버 라이드가 없으면 명령은 누산기상에서만 동작한다.

따라서, 명령세트는 우수한 조합을 한 길이의 명령세트이다.

1싸이클내에서 수행되는 1바이트길이의 일부 기본명령들이 있다.

프로그래머는 프리픽스명령으로 인해 명령의 기능을 확장할 수 있다.

명령은 프로그래머의 기호에 따라 8,16또는 24비트가 될 수 있다.

마리오프로세서는 고속인 내장캐시 ROM에 기억되어 있는 프로그램을 개시하는데 그리고 메모리에 지연되고 버퍼링된 입출력을 개시하는데 명령을 사용한 다.

그래픽처리는 상기 픽셀플롯 하드웨어를 사용하여 동작을 개시시키는 1싸이클 픽셀플롯명령을 사용함으로써 효율적으로 처리될 수 있다.

마리오명령세트를 설명하기 전에 명령 실행시 프로세서에 의해 세트되거나 액세스되는 여러메모리 맵핑 레지스터가 아래에 기술된다.

처음에 상태플래그 레지스터가 설명된다.

상태 레지스터는 16비트레지스터이며 레지스터 내에 각 16비트와 관련된 플래그가 아래에 설명된다.

[상태 플래그 레지스터(16비트)]

비트	플래그	
0	-	-
1	z	제로플래그
2	c	캐리플래그
3	s	부호플래그
4	v	오버플로우플래그([비트 14를 15로] XOR[15를 캐리로])
5	g	고(go)플래그 : 1 마리오칩작동, 0 정지
6	r	(R14)진행중인 ROM바이트페치
7	-	-

"고(GO)" 플래그 (비트 5)는 마리오칩이 동작중일 때는 "1" 상태로, 정지중일 때는 "0" 상태로 세트된다.

상기 마리오칩의 정지는 수퍼 NES에 제공될 인터럽트신호를 발생시킨다.

이 상태플래그의 상태는 수퍼 NES 프로세서가 체크한다.

비트 6은 ROM 바이트페치가 현재 진행중임을 나타낸다.

아래에 리스트된 게트(GET) 바이트 명령은 이 플래그가 클리어될 때까지 실행될 수 없다.

이 클리어는 데이타의 페치가 완료되었음을 나타낸다.

상태레지스터의 이들 최하위비트(LSB)는 마리오칩 프로세서 또는 주 CPU에 의해 잔여 8비트와는 독립적으로 또는 함께 리드(READ)된다.

상태플래그 레지스터의 최상위비트(MSB)는 소정의 프리픽스명령에 의해 세트되며 그리고 명령해독의 여러 모드를 설정한다.

비트	모드	
8	alt 1	변경 (ADD→ADC, SUB→SBC, ····)
9	alt 2	변경 (ADD→ADD#, SUB→SUB#, ····)
10	il	직접바이트 로 (ih전에 실행)
11	ih	직접바이트 하이 (hi가 준비될때까지 로바이트가 버퍼됨)
12	b	SReg & DReg, WITH에 의해 세트됨
13	-	-
14	-	-
15	irg	인터럽트플래그

상기 ALT 1모드에서, ADD 명령은 애드 위드 캐리(ADD WITH CARRY)로 해독되며, SUBTRACT 명령은 서브트랙트 위드 캐리(SUBTRACT WITH CARRY)로 해독된다.

명령 ALT 1은 이 모드에서 개시된다.

ALT 2명령은 ADD명령의 해독을 애드 위드 이미디어트 데이타(ADD WITH IMMEDIATE DATA)로, SUBTRACT명령의 해독을 서브트랙트 위드 이미디어트 데이타(SUBTRACT WITH IMMEDIATE DATA)로 변경한다.

"이미디어트(직접)" 데이타는 명령워를 바로 따르는 바이트를 말한다.

명령 ALT 3은 비트 8과 비트 9를 논리 "1" 로 세트시킴에 주의하자.

비트 10과 비트 11은 직접데이타가 직접 하이바이트냐 직접로바이트냐에 따라세트된다.

상태레지스터의 비트 12는 "b" 모드를 정의하는 것이며, 여기서 소스 레지스터와 행선지레지스터가 프리픽스명령 "WITH" 를 사용함으로써 세트된다.

상태 레지스터의 비트 15는 마리오칩이 동작을 정지한 후에 설정되는 마리오 인터럽트신호를 저장한다.

마리오칩은 상기 상태 레지스터 이외에도 다수의 레지스터를 포함하고 있다.

상기한 바와같이, 마리오칩은 제4(a)도 및 제4(b)도의 레지스터블록(76)을 논의할때 설명한 16비트인 16개의 레지스터를 포함하고 있다.

대부분의 이러한 레지스터들은 범용레지스터로서, 데이타 및 어드레스를 저장하는데 사용된다.

그러나 상기한 바와같이 레지스터(R15)는 항상 프로그램카운터로서 사용된다.

일반적으로 레지스터들은 2가지 목적으로 사용되는데 즉, 주 CPU와 통신하고, 실행중인 프로그램을 제어하는데 사용된다.

부가적으로 다른 레지스터들은 마리오칩내에서 사용되며 이들의 기능이 다음표에 설명되어 있다.

레지스터	특수기능
r0	디폴트 DReg 및 SReg
r1	PLOT 명령의 X 좌표
r2	PLOT 명령의 Y 좌표
r3	없다
r4	LMULT 명령결과의 하위워드
r5	없다
r6	FRMULT 및 LMULT 명령의 워드승수
r7	MERGE 명령의 소스1
r8	MERGE 명령의 소스2
r9	없다
r10	없다
r11	서브루틴 호출용 링크레지스터
r12	LOOP 명령의 계수
r13	루프명령이 분기되는 어드레스
r14	ROM 어드레스
r15	프로그램카운터
기타레지스터	
8비트 PCBANK	프로그램코드 뱅크레지스터
8비트 ROMBANK	프로그램데이타 ROM 뱅크레지스터(64K 뱅크)
8비트 RAMBANK	프로그램데이타 ROM 뱅크레지스터(64K 뱅크)
16비트 SCB	스크린베이스
8비트 NBP	비트플레인의 수
8비트 SCS	스크린 칼럼사이즈 선택 : 256, 320, 512, 640, 1024, 1280(2, 4 & 8비트 플레인에서 스크린 16 & 20 문자 하이)

마리오칩은 또한 칼라모드(CMODE) 레지스터를 포함하고 있다.

이 레지스터의 비트들 중 4비트는 후술될 특수효과를 낳는 예시적실시예에서 사용된다.

CMODE 레지스터 비트를 세팅함으로써 낳은 효과는 16 또는 256 칼라해상도 모드가 아래예의 설명대로 세트되었는지에 따라 변하게 된다.

CMODE 레지스터들은 다음과 같다.

[CMODE 비트 0]

· 플롯칼라 0 비트(NOT 투명비트)

· 16칼라모드에서 :

비트 0=1이고 선택된 칼라니블=0 이면 플로팅하지 마라.

· 256 칼라모드 및 비트 3=0에서 :

비트 0=1이고 칼라바이트=00이면 플로팅하지 마라.

· 256칼라모드 및 비트3=1에서 :

비트0=1 이고 칼라 하위니블=00이면 플로팅하지 마라.

· N.B.투명성 ON=0 투명성 OFF=1

· 투명성 OFF에 대한 사용만이 한 영역을 0, 즉 스크린을 클리어하는데 사용되는 0으로 채운다.

[CMODE 비트 1]

· 혼합비트

· 16칼라모드에서의 혼합(상위 니블 및 하위 니블은 두가지 칼라를 제공한다)

X위치 XOR Y위치 AND 1 = 0이면 하위 니블이 선택됨.

X위치 XOR Y위치 AND 1 = 1이면 상위 니블아 선택됨.

· 투명성이 유지되어 선택된 칼라 니블이 00이면 플로팅 하지 마라.

· 256칼라모드에서의 혼합은 무효이다.

[CMODE 비트 2]

· 상위 니블 칼라비트

· CMODE비트 3이 세트된 16칼라모드나 256칼라모드에서 :

이 비트(비트 2)가 세트되면 COLOUR명령은 칼라레지스터의 하위 니블을 소스바이트의 상위니블에 세트시킨다(이 상위니블은 또 다른 스프라이트의 하이 니블로서 저장된 18칼라 스프라이트를 언팩(unpact)하는데 사용된다).

· 칼라레지스터의 하위니블이 0이면 투명성이 0일지라도 플로팅 하지마라.

[CMODE 비트 3]

· 복잡한 비트

· 단지 258 칼라모드에서, 이 비트 3가 세트되면 칼라의 상위니블은 함부로 사용치 못하도록 제한되고 COLOUR 명령은 하위니블을 변경시킨다.

투명성은 하위니블만으로 계산된다.

· 보통의 256 칼라모드에서 투명도는 모든 비트를 사용하여 계산된다.

16칼라코드의 예

```
ibt     rO,$CO
colour                  ;   colour $CO을 세팅
ibt     rO,ZOOOO        ;   0으로 세트
cmode
ibt     rO,$97
colour
plot                    ;   colour $7을 플로팅
ibt     rO,$30
colour
plot                    ;   colour가 $0이므로 플로팅하지 않음
                        ;   (투명성 온 및 하위니블=0)
ibt     rO,ZCOOI        ;   비트 1을 세팅
cmode
ibt     rO,S40
colour
```

```
plot                ;   colour $0을 플로팅
                    ;   (투명성 오프)
stop
```

16칼라모드, 비트 2세트의 예

```
ibt     r0,$Co
Colour              ;   coulour $Co를 세팅
```

256칼라모드, 비트 3세트의 예

```
ibt     r0,$Co
colour              ;   colour $Co를 세팅
ibt     r0,%1000    ;   비트 3을 세팅
cmode
ibt     r0,$47
colour
plot                ;   colour $C7을 플로팅
ibt     r0,$50
colour
plot                ;   colour가 $Co이므로 플로팅하지 않음
                    ;   (투명성 온 및 하위니블=0)
ibt     r0,%1001    ;   비트 3 및 비트 1을 세팅
cmode
ibt     r0,$60
colour
plot                ;   colour $Co를 플로팅
                    ;   (투명성 오프)
stop
```

256칼라모드, 비트 3 및 비트 2세트의 예

```
ibt     r0,$Co
colour              ;   colour $Co세팅
ibt     r0,%1100    ;   비트 3 및 비트 2세팅
cmode
ibt     r0,$74
colour
plot                ;   colour $C7을 플로팅
ibt     r0,$03
colour
plot                ;   colour가 $Co이므로 플로팅하지 않음
                        (투명성 온 및 하위니블=0)
ibt     r0,%1101    ;   비트 3, 비트 2 및 비트 1을 세팅
cmode
ibt     r0,$08
colour
plot                ;   colour $Co를 플로팅
                    ;   (투명도 오프)
stop
```

다수의 마리오칩 레지스터들은 특수기능에 관계하고 있다.
상기 표에 나타낸 바와 같이 시스템은 달리 지정되지 않으면 특별한 명령에 필요한 행선지레지스터 또는 소스 레지스터로서 레지스터(R0)를 사용한다.
또한 레지스터(R0)는 ALU 누산기로서 사용된다.

상기 한 바와같이 곱셈 명령의 결과는 32비트이다.

그 32비트중 최하위 16비트들은 레지스터(R4)에 저장된다.

레지스터(R6)는 소수의 부호화된 곱셈명령(FRMOLT)과 긴곱셈 명령(LMULT)을 실행할때에 사용된다.

레지스터(R7)(R8)는 머지(MERGE)명령을 실행할때에 사용된다.

그 MERGE명령은 2개의 소정의 레지스터(즉, 레지스터(R7)(R8))를 사용하며 그리고 스프라이트 좌표데이타를 형성하기 위하여 그 두 레지스터를 함께 합병 (merge) 한다.

그러한 좌표데이타는 소정의 스프라이트를 소정의 다각형상에 맵핑시키기 위한 ROM테이블을 어드레싱하는데 사용된다.

이 MERGE명령은 다각형상에 맵핑된 스프라이트내에 포함될 다음 픽셀에 대한 칼라의 어드레스를 설정키위해 두 레지스터의 일부분들을 조합함으로써 텍스쳐 맵핑동작을 효율적으로 수행하는데 도움이 된다.

레지스터(R11,R12,R13)는 서브루틴의 실행을 제어하는데 사용된다.

레지스터(R11)는 서브루틴 호출용 링크레지스터로서 사용되며 프로그램 카운터의 내용에 1을 가산한 결과치를 저장한다.

레지스터(311)의 내용은 루프가 완료된후에 액세스되어야하는 어드레스를 나타낸다.

레지스터(RI2)는 루프의 반복횟수를 나타내는 숫자를 저장하는데 사용된다.

루프의 어드레스는 레지스터(R13)에 저장된다.

상기한 바와같이 레지스터(R12)의 내용이 변경될 때마다 이 레지스터(R13)내에 저장된 어드레스에 있는 하나의 바이트가 ROM(10)으로부터 리드된다.

이 방법으로 지연된(또는 버퍼된) READ동작은 후술될 게트바이트 명령과 함께 실행된다.

상기 테이블의 "기타 레지스터" 를 다시 볼것 같으면, 프로그램 ROM에서 프로그램이 실행되고 있는 위치는 24비트 어드레스로 어드레싱된다.

이 지정된 어드레싱 최하위 16비트들은 프로그램카운터가 가지고 있다.

프로그램뱅크를 설정하는 최상위비트들은 프로그램 코드뱅크(PC뱅크) 레지스터에 저장되어 있다.

ROM뱅크레지스터(ROMBANK)는 마리오칩 프로세서가 ROM(10)에 저장되어 있는 프로그램데이타를 어드레싱하여야 할 비트중 최상위비트를 저장하고 있고, 그리고 이 ROM뱅크레지스터의 비트는 레지스터(R14)에 저장되어 있는 16비트 ROM어드레스비트에 추가된다.

유사하게, RAM뱅크레지스터(RAMBANK)는 마리오칩 프로세서가 액세스하고자 하는 RAM의 프로그램데이타의 어드레스비트들을 저장하고 있다.

이들 RAM뱅크레지스터 및 ROM뱅크레지스터의 내용은 마리오칩 프로세서의 어드레싱범위를 효과적으로 확장하는데 사용되는 마리오칩의 ROM 및 RAM 어드레싱 명령과 관련하여 사용된다.

스크린베이스 레지스터(SCB)는 여기서 생성 및 회전 그리고 확대 또는 축소될 스프라이트 또는 대상체의 가상비트맵의 어드레스를 저장하는데 사용된다.

PLOT픽셀명령이 실행될때, 스크린베이스 레지스터(SCB)는 정보가 액세스 되거나 라이트될 RAM의 어드레스를 저장한다.

레지스터(NBP)는 사용중인 비트플레인의 갯수를 저장하는데 사용된다.

전형적인 레지스터(NBP)가 나타내는 비트플레인은 2비트플레인, 4비트플레인 또는 8비트플레인중의 하나이다.

부가적으로, 스크린칼럼사이즈 레지스터(SCS)는 칼럼에 포함된 문자의 수효에 기초하여 수직비트맵에 대한 정보를 나타내는데 사용된다.

마리오칩의 명령세트는 아래에서 기술되는데, 즉 관련명령을 디코딩함으로써 수행되는 명령니모닉 및 관련기능이 기술된다.

먼저, 설명을 요하지 않는다는데 의문을 일으키는 명령의 특수기능에 대한 설명이 선행된다.

스톱(STOP)명령은 마리오칩이 그의 동작을 완료했을때 실행되어 "고(GO)" 플래그를 0으로 세트시킴과 동시에 인터럽트신호를 발생시켜 주 CPU에 제공한다.

캐시(CACHE)명령은 마리오칩의 캐시RAM에 메모리될 프로그램 ROM의 일부분을 설정하는데 사용된다.

캐시명령이 실행되면, 프로그램카운터의 내용이 캐시 베이스 레지스터로 전송됨과 동시에 후술될 캐시태그가 리세트된다.

마리오칩은 분기명령 다음의 명령이 아래표와 같이 실행되는 일련의 지연분기 명령을 포함하고 있다.

분기가 발생되는 어드레스는 프로그램카운터의 내용과 관련되어 있다.

명령세트는 아래표의 조건에 따른 다양한 지연분기를 포함하고 있다.

마리오칩은 다수의 "프리픽스" 명령 즉 투(to), 위드(with)및 프럼(from)을 포함하고 있다.

이들 프리픽스명령은 다음 명령들에 대한 자료분배를 내포하고 있다.

예를들면, "TO" 프리픽스는 다음 명령에게 행선지레지스터(DReg)를 설정해준다.

"FROM" 프리픽스는 다음 명령에게 소스레지스터(SReg)를 설정해준다.

"WITH" 프리픽스는 상기 두 설정기능을 모두 가지고 있다.

대부분의 명령은 옵코드(opcode)에서 제 2소스레지스터라 불린다.

그러나, 소스레지스터와 행선지 레지스터가 프리픽스명령에 의해 설정되지 않으며 레지스터(R0)가 소스레지스터 및 행선지레지스터로 사용된다.

프로그램 카운터인 R15가 행선지레지스터로 설정되어 다음 명령으로하여금 R15에 그의 내용을 저장하게 함으로써 1싸이클 지연분기가 개시된다.

다른 프리픽스명령들은 다음 명령의 동작을 변화시키기 위해 상태 레지스터의 상위 바이트에 있는 플래그를 세팅시킨다.

모든 비 프리픽스명령은 상태워드의 상위 바이트를 클리어시킨다.

다음은 프리픽스명령을 통해 다음 명령이 어떻게 변경되는가에 대한 예이다.

```
1 sr                          ; rG = rO를 1만큼 우측이동
to r2 1sr                     ; r4 = rO를 1만큼 우측이동
from r4 1sr                   ; rO = r4를 1만큼 우측이동
alt 1 from r6 to r5 add r7    ; rE = r6 + r7 + 캐리
alt 1 with r3 add r3          ; r3 = r3 + r3 + 캐리(6502ro1)
```

만일 "6" 플래그가 상태 레지스터에서 설정되면 "10" 명령은 "MOVE" 명령으로서 동작하도록 변경되어지게 된다.

TO명령은 정보가 이동되어야하는 행선지 레지스터를, FROM명령은 정보소스를 각각 나타낸다.

STW명령이 버퍼내에 특수워드를 저장하고 있으므로 다음 명령을 실행하기에 앞서 저장동작이 완료되기를 기다릴 필요가 없다.

이 때문에 프로세서보다 처리속도가 떨어지는 RAM을 사용한다는 것이 프로세서의 처리속도를 쓸데없이 더디게 하지 않는다.

루프(LOOP)명령의 실행 이 그 반복을 거듭할수록 루프가 반복되어야할 횟수를 기억하고 있는 범용레지스터(R12)의 내용은 감소한다.

범위 레지스터(R12)의 내용이 0이 아니면 레지스터(R13)에 의해 지정된 어드레스로 점프가 개시된다.

Alt 1, Alt 2 및 Alt 3는 실행된 명령이 아래표에 나타낸 바와같이 해독되도록 하기 위하여 상태 레지스터의 상기 플래그를 세팅시키는 프리픽스 명령들이다.

PLOT명령은 플로팅될 픽셀의 X 및 1스크린 좌표를 식별하고, 레지스터(R1)(R2)로 나타낸 X 및 Y좌표에 따른 스크린위치에 COLOR명령에 의해 지정된 칼 라를 플로팅한다.

PLOT픽셀명령을 실행하면 수평라인을 고속으로 플로팅하는데 도움이 되도록 레지스터(R1)의 내용이 자동적으로 증가하고, 그리고 여분의 증가된 명령을 포함하지 않게 된다.

Alt 1플래그가 세팅되면 PLOT명령은 READ PIXEL명령(RPIX)으로서 해석된다.

상기 리드픽셀명령은 실행시킴으로써 지정된 스크린위치에 있는 픽셀의 칼라이 판독되어 픽셀하드웨어로부터 원하지 않는 픽셀정보를 플러시하는데 사용된다.

본질적으로 리드픽셀 명령은 명령에서 지정된 특정한 픽셀의 칼라를 결정하기 위하여 문자매트릭스로부터의 리드와는 반대로 플롯하드웨어를 사용한다.

COLOR명령은 지정된 소스레지스터의 내용에 의해 정해진 다음 픽셀의 색깔을 칼라 하드웨어에 제공한다.

"CMODE" 명령은 칼라모드를 설정하며 그리고 상기 예제들에서 설명된 다른 특수효과를 발생시키게 사용될 수 있다.

예를들면, 이 CMODE명령을 사용하여 채색효과를 생성하기 위해 다른 픽셀에 서로다른 칼라를 교대로 오게하여 혼합효과를 낳을 수 있다.

또한 CMODE명령은 투명성(transparency)을 제어하는데 사용될 수 있으므로 스프라이트의 디스플레이는 후면디스플레이를 차단할 수 있다.

이 투명성은 상기 예제들에서 보여진 칼라모드 관련플래그를 설정함으로써 정해진다.

또한 명령세트는 디스플레이될 대상체의 기울기를 정하기 위해 다각형을 회전시키는데 대한 계산을 할때 사용되는 부호화된 소수부 곱셈을 포함한다.

증가명령은 레지스터(R14)와 함께 사용되면 ROM으로부터의 리드동작을 개시 시킬것이다.

게트 C명령은 ROM으로부터 액세스된 바이트를 칼라레지스터르 로드시킨다.

다음의 표는 이제까지 기술해왔던 명령을 포함하고 있는 본 발명의 한 실시예에 따른 예시적인 마리오칩 명령

세트를 나타낸다.

명령세트

16진수	니모닉	기 능
$00	STOP	마리오칩정지, 65816 IRQ g=0
$01	NOP	1싸이클 미작동
$02	CACHE	캐시베이스를 PC에 설정, 캐시플래그를 리세트시킴(단, CP가 현재 캐시베이스와 다름) : 캐시베이스< > r15이면 캐시베이스는 캐시플래그를 리세트시킨다.
$03	LSP	논리시프트라이트 : DReg = SReg LSR 1
$04	ROL	캐리와 함께 로테이트시프트 : DReg = SReg ROL 1
$05nn	BRA sbyte	지연분기 : r15 = r15 + 부호바이트오프셋
$06nn	BGE sbyte	크거나 같으면 지연분기 : 조건(sXORv) = 1, r15 = r15 + 부호바이트 오프셋
$07nn	BLT sbyte	작으면 지연분기 : 조건(sXORv) = 0, r15 = 15 + 부호바이트 오프셋
$08nn	BNE sbyte	같으면 지연분기 : 조건 z = 1, r15 = r15 + 부호바이트오프셋
$09nn	BEQ sbyte	같지 않으면 지연분기 : 조건 z = 0, r15 = r15 + 부호바이트오프셋

$0ann	BPL sbyte	양(+)이면 지연분기 :
		조건 s = 0, r15 = r15 + 부호바이트오프셋
$0bnn	BMI sbyte	음(-)이면 지연분기 :
		조건 s = 1, r15 = r15 + 부호바이트오프셋
$0cnn	BCC sbyte	캐리클리어되면 지연분기 :
		조건 c = 0, r15 = r15 + 부호바이트오프셋
$0dnn	BCS sbyte	캐리가 세트되면 지연분기 :
		c = 1이면 r15 = r15 + 부호바이트오프셋
$0enn	BVC sbyte	오버플로우가 클리어되면 지연분기 :
		v = 0이면 r15 = r15 + 부호바이트오프셋
$0fnn	BVS sbyte	오버플로우가 세트되면 :
		v = 1이면 r15 = r15 + 부호바이트오프셋
$10-$1f	To r0..r15	(프리픽스) DReg를 rn으로 세팅
		(다음 옵코드용 명선지 레지스터)
ifb :	MOVE	n = SReg(어느 플래그도 세팅안됨)
$20-$2f	WITH r0...r15	(프리픽스) DReg & SReg를 rn으로 세팅
		(소스플래그 & 명선지플래그 & b플래그) :
		DReg = rn
		SReg = rn
		b = 1
$30-$3b	STW(rn)	SReg를 어드레스(rn)에 저장 :
		RAM(rn) = SReg(워드로/하이버퍼)
		(보통 짝수어드레스상의 워드)
if alt 1 :	STB(rn)	SReg의 로바이트를 어드레스(rn)에 저장 :
		RAM[rn] = SReg 1 (바이트버퍼)
$3c	LOOP	감소 r12 및 r12 < > 0이면 어드레스(r13)
		으로 지연점프 :

		r12 = r12-1
		r12 < > 0 이면 r15 = r13
$3d	ALT 1	(프리픽스) alt 1플래그세팅 :
		alt 1 = 1
$3e	ALT 2	(프리픽스) alt 2플래그세팅 :
		alt 2 = 1
$3f	ALT 3	(프리픽스) alt 1플래그 및 alt 2플래그
		세팅 :
		alt 1 = 1
		alt 2 = 1
$40-$4b	LDW(rn)	어드레스(rn)로부터 DReg로딩 :
		DReg = RAM[rn](워드 로/하이 대기)
		(보통 확수어드레스상의 어드레스)
if alt 1 :	LDB(rn)	어드레스(rn)로부터 DReg로딩 :
		DReg 하이 = 0
		DReg 로 = RAM[rn](바이트 대기)
$4c	PLOT	픽셀을 r1,r2(x,y)에 플로팅 그리고 증가
		r1(N.B.r1 및 r2는 스크린상에 있는지
		체크안됨, RAM의 아무데나 작성) :
		플롯 (r1,r2)
		r1 = r1 + 1
if alt 1 :	RPIX	r1,r2(x,y)에서 픽셀의 칼라판독 :
		DReg = 점(r1,r2)
$4d	SWAP	바이트교체 :
		DReg 로 = SReg 하이
		DReg 하이 = SReg 로
$4e	COLOUR	PLOT칼라세팅 :

		플롯칼라 = SReg
if alt 1 :	CMODE	PLOT칼라모드세팅 :
		플롯칼라모드 = SReg
$4f	NOT	DReg = NOT SReg
$50-$5f	ADD r0...r15	DReg = SReg + m
if alt 1 :	ADC	DReg = SReg + m + c
if alt 2 :	ADD	DReg = SReg + #n
if alt1+alt2 :	ADC	DReg = SReg + #n + c
$60-$6f	SUB r0...r15	DReg = SReg - m
if alt 1 :	SBC	DReg = SReg - m - c
if alt 2 :	SUB	DReg = SReg - #n
if alt1+alt2 :	CMP	SReg - m (제로, 부호, 캐시, 오버플로우)
$70	MERGE	r7 및 r8의 하위바이트를 DReg로 합병 :
		DReg.h = r7.h
		DReg.l = r8.h
		플래그세트결과 :
		s = b15 OR b7
		v = b14 OR b6 OR s
		c = b13 OR b5 OR v
		z = b12 OR b4 OR c
$71-$7f	AND r1...r15	DReg = SReg AND m
if alt 1 :	BIC	DReg = SReg AND NOT m
if alt 2 :	AND	DReg = SReg AND #n
if alt1+alt2 :	BIC	DReg = SReg AND NOT #n

$80-$8f	MULI r0...r15	DReg = SReg * Rn (부호 8×8비트)
if alt 1 :	UMULI	DReg = SReg * Rn (비부호 8×8비트)
if alt 2 :	MULI	DReg = SReg * #n (부호 8×8비트)
if alt1+alt2 :	UMULI	DReg = SReg * #n (비부호 8×8비트)
$90	SBK	SReg를 사용된 바로전의 RAM어드레스에 저장
$91-$94	LINK 1...4	복귀어드레스를 r11에 연결
		r11 = r15 + 1...4
$95 :	SEX	워드의 확장로 바이트에 부호를 :
		DReg.[b15-b7] = SReg.[b7]
		DReg.1 = SReg.1
$96	ASR	산술시프트라이트 :
		DReg = SReg ASR 1
if alt 1 :	DIV 2	반올림 없이 2로 나눔 :
		DReg = SReg ASR 1
		(DReg = -1이면 DReg = 0)
$97	ROR	캐리와 함께 우측 시프트 :
		DReg = SReg ROR 1
$98-$9d	JMP r8...r13	어드레스(rn)로 점프 :
		r15 = rn (지연분기)
if alt 1 :	LJMP	어드레스(rn)으로 긴 점프(SReg로부터
		ROM뱅크)
		그리고 케시리세트 :
		r15 = rn (지연분기)
		프로그램 ROM뱅크레지스터 = SReg
$9e	LOB	하위바이트 :
		DReg 상위바이트 = 0

		DReg 하위바이트 = SReg 하위바이트
$9f	FMULI	소수부호곱셈 :
		DReg = (sSReg*r6)
		(부호 16×16비트곱셈)
		c = (SReg)*r6
if alt 1 :	LMULI	긴 부호곱셈 :
		DReg = (SReg*r6).hw
		(부호 16×16비트곱셈)
		r4 = (SReg*r6).hw
		c = (SReg)*r6).b15
$a0-$afnn	IBI r0...r15,sbyte	부호확장바이트를 rn에 로딩 :
		rn = 직접바이트(부호확장)
if alt 1 :	LMS r0...r15, byte	절대시프트바이트 어드레스로부터
		rn을 로딩 :
		rn = RAM[byte <<1] (워드데이타)
if alt 2 :	SMS r0...r15, byte	절대시프트바이트 어드레스에 rn을
		기억 :
		RAM[byte <<1] = rn(워드데이타)
$b0-$bf	FROM r0...r15	SReg = rn세팅 :
		SReg = rn
ifb :	MOVES	DReg = rn
		(제로플래그,부호 및 오버플로우플래그)
$c0	HIB	상위바이트 :
		DReg.h = 0
		DReg.l = SReg.l
$c1-$cf	OR r1...r15	DReg = SReg OR Rn
		DReg = SReg XOR Rn
		DReg = SReg OR #7
		DReg = SReg XOR #7
$d0-$de	INC r0...r14	rn증가 :
		rn = rn + 1
		(TO/WITH/FROM은 무시)
$df	GETC	ROM버퍼로부터 PLOT칼라 R로 바이트를
		얻음
if alt 2 :	RAMB	RAM데이타 뱅크레지스터 = SReg
if alt1+alt2 :	ROMB	ROM데이타 뱅크레지스터 = SReg
$e0-$ee	DEC r0...r14	rn감소 :
		rn = rn-1
		(TO/WITH/FROM은 무시)
$ef	GETB	ROM버퍼로부터 Dreg로 부호화바이트
		GET :
		DReg = ROM버퍼바이트.
		제로확장
if alt 1 :	GETBH	ROM버퍼로부터 Dreg로 상위바이트를
		GET :
		DReg = ROM버퍼바이트.
		DReg = (SReg & $FF) + (byte <<8)
if alt 3	GETBL	ROM버퍼로부터 Dreg로 하위바이트를
		GET :
		DReg = ROM brff.byte.
if alt1+alt2 :	GETBS	ROM버퍼로부터 Dreg로 부호바이트
		GET :
		DReg = ROM buff.byte
$f0-$ffnnnn	IWT ro...r15, word	직접워드를 rn에 로딩 :
		rn = 직접워드(버퍼)
if alt 1 :	LM r0...r15, word	절대워드 어드레스로부터 rn로딩 :
		rn = RAM[워드어드레스](워드데이타)
if alt 2 :	SM r0...r15, word	절대워드 어드레스를 rn에 저장

제6도 내지 제17도에는 제4(a)도와 제4(b)도에서 블럭다이어그램으로 나타낸 구성부가 더 상세하게 나타내

어져 있다.

본 발명의 특징을 보다 명료하게 나타내기 위하여, 이 분야에서 통상의 지식을 가진자가 명백히 알수 있는 사항과 본 특징을 애매하게하는 설명들은 도면에 도시하지 않았다.

제4(a)도에 도시된 ALU(50)가 제6도에 보다 상세히 도시되어 있다.

제4(a)도와 제6도에 도시된 바와같이 ALU(50)는 X버스, Y버스 및 Z버스와 연결된다.

마리오칩의 범용레지스터(R0~R15)는 ALU와 연결된다.

ALU(50)는 16비트 가감산기(152)를 이용하여 가감산을 수행한다.

또한 ALU(50)는 종래의 "ALD" 논리회로(152)와 "OR" 논리회로(156)와 "XOR" 논리회로(158)를 포함하고 있다.

또한, ALU(50)는 종래 시프트기능회로를 포함하고 있기때문에 임의 캐시가 라인(160)을 통해 멀티플렉서(164)의 한쪽 입력에 제공되기전에 MSB위치로 시프트된다.

부가적으로, ALU(50)는 종래의 바이트교체기능, 즉, 버스상의 LSB와 MSB가 라인(162)을 통해 멀티플렉서(162)에 제공되기 전에 교체되어지는 바이트교체기 능을 수행한다.

X버스와 Y버스는 제 6도에 도시된 바와같이 여러회로(152,154,156,158)에 연결된다.

가감산기(152) 및 회로(154,156,158)의 각 출력, 시프트기능출력 및 교체기능출력은 모두 6입력-1출력인 16비트 6 ×1멀티 플렉서(MUX)(164)에 연결된다.

디코딩된 명령에 따라 적절한 결과가 행선지 버스(Z)상으로 출력된다.

가감산기(152)는 X버스로부터 16비트를 수신하는 것 이외에, Y버스상의 정보 또는 명령자체내의 정보들 중의 하나를 멀티플렉서(50)에 입력된 명령 디코더의 입 력에 기초하여 선택수신한다.

부가적으로 ALU(50)는 CPU플래그회로(166)를 포함하고 있다.

이 CPU 플래그회로(166)는 제로오버플로우신호, 부호신호 및 캐리신호를 발생시켜 CPU플래그회로(166)내에 있는 적어도 하나의 플래그레지스터로 로딩시킨다.

CPU플래그들은 명령디코딩회로(60), 즉 명령(플래그를 가감산기(152)에 의해 결정된 조건에 따라 세팅시키는 명령)에 의해 발생된 캐리인에이블신호, 제로인에이블신호, 부호인에이블신호 및 오버플로우 인에이블신호를 디코딩하는 명령 디코딩회로(60)에 의해 세팅된다.

또한 그 CPU플래그들은 행선지 버스(Z)상으로 CPU플래그회로(166)에 입력된 내용에 따라서도 세팅된다.

가령, 플래그들은 다양한 조건에 따라 조건분기동작을 개시시키는데 사용된 다.

제7도, 제8(a)도 및 제8(b)도에는 제4(a)도에 도시된 픽셀플롯회로(52,54,56,58)가 더욱 상세하게 도시되어 있다.

이 픽셀플롯회로는 지정된 X좌표와 Y좌표를 갖는 PLOT명령을 실행하여 COLOR명령에 의해 로딩된 칼라레지스터(54)의 내용에 따라 지정된 칼라로 그들 스크린좌표상에 픽셀을 플로팅한다.

상기한 바와같이, 수퍼 NES는 문자맵 디스플레이스크린을 사용한다.

플롯하드웨어는 픽셀좌표 어드레스데이타를 문자맵 어드레스데이타로 변환시킨다.

수퍼 NES문자는 비트플레인으로 설정된다.

문자들은 4개 또는 16개 또는 256개의 칼라를 설정하기 위한 2비트 또는 4비트 또는 8비트플레인을 가질 수 있다.

문자설정의 각 바이트는 문자의 1픽셀행의 비트플레인을 포함하고 있다.

픽셀들은 좌에서 우로, 즉 상위 비트에서 하위 비트로 설정된다.

그리고 256칼라모드에 대해서는 갱신되어야할 8RAM위치가 존재하고 있다.

픽셀플롯 하드웨어는 모든 비트가 궁극적으로 갱신될 필요가 있으므로 디스플레이될 특정 바이트내의 그러한 모든 비트를 저장하는 칼라매트릭스(206)를 포함하는 국부버퍼 메카니즘을 포함하고 있다.

그리고 그 칼라매트릭스회로(206)에 비트플레인카운터(208)가 연결되어 있다.

픽셀좌표들은 X버스와 Y버스로부터 플롯 X레지스터(202)와 플롯 Y레지스트(204)로 로딩된다.

본 실시예에서는 범용레지스터(R1)(R2)가 제7도의 플롯 X레지스터(202)와 플롯 Y레지스터(204)로 각각 사용된다.

이들 레지스터(202)(204)는 PLOT명령에 의해 지정, 작성될 픽셀의 X좌표와 Y좌표를 수신한다.

플롯 X레지스터(202) 및 플롯 Y레지스터(204)는 그 위치 배럴시프트회로(2 position barrel shifting circuit)(214)로 어드레스를 출력시키는 기능을 갖는 전가산기(FA) 및 반가간기(HA)에 기초한 문자어드레스 계산회로에 연결된다.

차례로, 그 2위치 배럴시프트회로(214)는 플롯어드레스 레지스터(216)와 어드레스비교기(218)에 연결된다.

플롯 X레지스터(202)의 3개의 최하위 비트들은 3 ×8 디멀티플렉서(212)에 연결되고, 이 디멀티플렉서(212)는 비트펜딩 회로(210)에 연결된다.

제8(a)도에 도시된 플롯제어회로(200)는 후술될 다른 제어신호뿐만 아니라 PLOT픽셀(PLOT)명령이나 READ픽셀(RPIX)명령이 디코드되었음을 지시하는 신호를 수신한다.

플롯제어회로(200)는 다음에 기술될 플롯회로 제어신호들을 발생시킨다.

상기한 바와같이 플롯제어회로(200)는 픽셀플롯 하드웨어(52)내에서 사용될 제어신호들을 발생시킨다.

제8(a)도에 나타낸 바와같이 픽셀제어회로(200)는 AND게이트(201)를 통해 연결된 비트펜딩 레지스터(210)의 출력을 수신한다.

이 비트펜딩 레지스터(210)의 8비트 모두가 세트되면 픽셀제어회로(200)는 판독싸이클이 스킵되어 칼라메트

릭스회로(206)내의 정보가 RAM에 기록되었음을 알게 된다.

픽셀제어회로(200)는 또한 동작을 개시하기 위해 PLOT 명령에 응답한다.

픽셀제어회로(200)는 또한 새로운 정보가 칼라매트릭스회로(206)(정보를 RAM으로 출력시킴)에 기록되지 않는다는 점만을 제외하면 사실상 동일한 동작을 개시시키기 위해 리드픽셀명령(RPIX)에 응답한다.

상기한 바와같이 리드(READ)픽셀명령은 스크린상에 있는 특정픽셀의 칼라를 알 필요가 있을경우에만 실행되고, 또한 칼라매트릭스회로(206)에서 존재하는 정보를 플러시(flush)하는데 사용된다.

픽셀제어회로(200)는 또한 RAM액세스가 완료되었음을 나타내는 RAM완료 제어신호(RAM DONE)를 수신한다.

상기한 바와같이 RAM완료제어신호는 칼라매트릭스회로(206)내의 비트플레인을 식별하는 비트플레인카운터(208)을 증가시키는데 사용된다.

또한, 플롯제어회로(200)는 어드레스비교기(218)로부터 PLEQ신호를 수신하는데, 이 신호는 어드레스매치가 있었음을 그리고 칼라매트릭스회로(206)의 내용을 RAM에 기록할 필요가 없음을 지시하여, 현재의 칼라매트릭스회로(206)의 내용에 대해 갱신이 계속되어야함을 지시한다.

플롯제어회로(200)는 또한 얼마나 많은 바이트가 판독 및 기록되어야하는지를 플롯제어회로(200)에게 알리는 스크린모드(SCB.MD)제어신호를 수신한다.

플롯제어회로(200)는 칼라매트릭스회로(206)의 내용이 제 2버퍼부에서 버퍼되도록 지시하는 제7도 및 제8(b)도의 덤프제어신호(DUMP)를 발생시킨다.

플롯제어회로(200)는 부가적으로 클리어 비트펜딩 레지스터신호(CLRPND)와 로드비트 펜딩 레지스터 제어신호(LDPND)를 발생시켜 비트펜딩 레지스터(210)에 제공한다.

부가적으로 플롯제어회로(200)는 제8(b)도에서 기술한 칼라매트릭스회로의 구성요소들과 관련된 LDPIX제어신호와 BPR제어신호를 발생시킨다.

명령 디코더로 PLOT명령을 디코딩하고 PLOT신호를 플롯제어회로(200)에 입력시키면, 픽셀플롯 하드웨어가 달리 바쁘지 않다는 조건하에서, 로드펜딩 신호(LDPND)의 발생이 개시된다.

이 LDPND신호는 디멀티플렉서(212)로부터 비트펜딩 레지스터(210)로 데이타를 로딩시키기 위하여 비트펜딩 레지스터(210)에 제공된다.

클리어 펜딩신호(CLRPND)는 펜딩데이타가 RAM에 기록되었음을 지시하는 RAM완료신호(RAM DONE)에 응답하여 발생된다.

그후 비트휀딩 레지스터는 다음 픽셀플롯명령에 대해 자유로워진다.

플롯제어회로(200)가 수신한 신호와, 여러가지 어드레스신호 및 데이타 신호와, 다른 관련제어신호와, 플롯제어회로에서 발생된 출력제어신호와의 관계를 나타내는 타이밍도가 제8(c)도에 도시되어 있다.

마찬가지로 예시적인 어드레스값, 데이타값 등이 단지 설명을 위해 도시되 어 있다.

플롯하드웨어(152)는 다음과같이 동작한다.

플롯하드웨어(152)가 사용중이 아님을 플롯제어회로(200)가 결정하면 제4(a)도의 칼라레지스터(54)의 내용이 8×8칼라매트릭스회로(206)의 수평행으로 로드 된다.

칼라매트릭스(200)는 행(raw)을 단위로하여 로딩되고 열(column)단위로 판독 된다.

칼라레지스터(54)의 내용은 COLOR명령에 의해 갱신된다.

다음 PLOT명령은 이 칼라레지스터(54)를 통해 칼라데이타를 칼라매트릭스 회로로 로드시킨다.

칼라레지스터의 비트가 로드될 장소인 칼라매트릭스회로(206)내의 수직위치를 플롯 X레지스터(202)내에 저장된 3개의 LSB에 의해 결정된다.

따라서 플로팅될 어드레스의 3개의 LSB는 칼라매트릭스회로(206)에서 갱신될 비트의 행을 결정한다.

비트펜딩레지스터(210)는 스크린문자의 부분중의 어느 특정비트가 갱신되고 있는지를 기록하는데 사용된다.

비트펜딩레지스터(210)는 비트들이 스크린의 관련부분에 라이트되고 있음을 지시하기 위한 16개의 레지스터 플래그를 가지고 있다.

비트펜딩 레지스터(210)는 LDPND신호에 응답하여 로드되고, 플롯제어기(210)에 의해 발생된 CLRPND신호에 의해 클리어된다.

만일 다음 플롯명령이 동일 영역에서 스크린맵을 갱신하기 위해 실행되면 주어진 비트에 대한 동작이 8×8칼라매트릭스회로(206)로 로딩된 픽셀에 따른 부가적인 칼라데이타와 함께 반복된다.

또 다른 비트는 플롯 X레지스터(202)에 저장된 플롯어드레스의 LSB를 통해 비트펜딩 레지스터(210)로 세팅된다.

특수비트는 플롯 X레지스터(202)와 연결된 3 ×8 디멀티플렉서(212)를 통해 비트펜딩레지스터로 로딩된다.

갱신될 픽셀이 수평으로 8픽셀이상 떨어져 있으면 또는 그 픽셀이 다른 수직위치를 차지하고 있으면 칼라매트릭스회로(206)에 기록되어질 데이타는 RAM(6 또는 8)로부터 판독되어야만 한다.

그후 칼라매트릭스회로(206)는 새로운 칼라데이타를 자유로이 수신할 수 있다.

칼라매트릭스회로(206)의 현재 내용은 RAM에 기록하라는 다음의 플롯명령이 수신될때까지 픽셀플로터 하드웨어, 예를들면 칼라매트릭스회로(206)내에 버퍼된다.

칼라매트릭스회로(206)의 데이타가 RAM(6)또는 RAM(8)에 기록되면, X,Y 좌표를 RAM어드레스로 변환시키기 위한 어드레스변경 계산이 논리게이트, 예를들면 제7도의 전가산기 및 반가산기를 사용하여 행해진다.

그 실제어드레스계산은 후술될 설명 및 예시적코드에 따라 행해진다.

그러한 계산은 4,16 또는 256칼라모드가 사용되느냐에 따라 바뀌게 된다.

계산예에서는 256칼라모드를 사용한다.

이들 256 칼라문자는 각각 64바이트 전체에 대해 비트플레인의 쌍을 설정하는 16바이트의 4블록을 가지고 있다.

비트맵은 원하는 스크린영역의 모든 위치에 특정문자를 배치시킴으로써 구성된다.

그리고 수퍼 NES를 사용하여 플로팅할때에는 가로줄, 즉 열(column)에 문자를 작성하는 것이 바람직하다.

예) (128픽셀 하이스크린(high screen))

문자수			
0	16	32	.....
1	17	33	
2	18	24	.....
.	.	.	
.	.	.	
15	31	47	.....

비트맵사이즈는 메모리와 DMA 전달시간에 의해 주로 제약을 받는 반면 수퍼 NES에 의해서는 제약을 받지 않는데, 그 이유, 즉 수퍼 NES에 의해서는 제약을 받지 않는 이유는 수퍼 NES가 플로팅할 수 있는 문자가 256 문자에 한정되지 않기 때문이다.

예를들면 마리오칩은 128 및 160 픽셀 하이스크린상에 문자플로팅할 수 있다.

최대 스크린폭은 32문자, 즉 256픽셀이다.

다음의 알고리즘은 픽셀을 플로팅하는 동작이 칼럼에 작성된 가상비트맵을 사용하여 어떻게 제어되는가를 예시하고 있다.

먼저, X 좌표의 최하위 3비트를 이용하여 모든 비트플레인에 대한 픽셀 마이크를 계산한다.

픽셀수	마스크
0	%10000000
1	%01000000
.	.
.	.
7	%00000001

다음에 칼럼아래에 문자를 제공하기 위하여 하위 3개의 비트가 제거된 y 좌표를 사용하여 칼럼 아래의 오프셋을 계산하고, 이어서 문자의 사이즈를 곱한다.

스크린칼라	바이트의 문자사이즈
4	16
16	32
256	64

다음에 하기 3개의 비트가 제거된 X 좌표를 이용하여 문자칼럼 위의 오프셋을 계산한다.

여기에서 칼럼사이즈는 칼럼에 작성된 문자들의 갯수에다 문자사이즈를 곱한 것이다.

보통의 칼럼사이즈

문자상위			
	16	20	
4	256 바이트	320 바이트	
칼라	16	512 바이트	640
256	1024 바이트	1280 바이트	

y 좌표의 하위 3개비트는 문자아래에 바이트오프셋을 제공한다.

모든 오프셋과 현재의 비트맵에 대한 포인터를 합한 것은 픽셀의 제1비트 플레인을 구성하고 있는 바이트의 어드레스를 제공한다.

다음의 비트플레인은 차례로 1바이트이고 그 이후 마지막으로부터 15바이트이다.

픽셀비트는 픽셀마스크를 사용하여 세트되거나 클리어될 수 있다.

각 비트플레인의 비트는 픽셀이 요구하는 칼라레지스터(54)에 저장되어 있는 칼라의 수에 대응하는 비트의 상태로 세트되거나 클리어된다.

예제코드(EXAMPLE CODE)

; 본 발명의 게임설명에서 사용된 바와 같이 65816코드씩 4비트플레인상에 플로팅.

; 루틴은 대부분 표로 도시됨

```
; 레지스터 A, X 및 Y는 16비트.
칼라설정(Set Coulour)
; 칼라 및 더블(double)을 얻음.
lda         colour
asl         a
tax
; 비트플레인 0과 1에 칼라마스크 설정.
lda         mask1 tab, X
sta .       mask1
; 비트플레인 2와 3에 칼라마스크 설정.
lda         mask1 tab, X
sta .       mask2
rts
플롯
; 수평좌표와 수직좌표를 얻음.
; 상기 두 좌표를 배가시켜 Y 레지스터와 X 레지스터로 이동.
lda          plotx1
asl          a
tay          ; Y 는 X 좌표 * 2
lda          ploty1
asl          a
tay          ; x 는 y 좌표 * 2
 ; 칼럼아래의 오프셋을 얻음.
lda          pyoftab,x
; 칼럼오프셋의 개시점을 가산.
clc
adc          pxoftab, y
; 이중버퍼포인터(비트맵선택)
clc
adc         drawmap
tax
; X 는 비트맵베이스로부터 원하는 픽셀을 갖기 위한 워드의 오프셋
; Y는 픽셀의 X 좌표 * 2
; 비트플레인 0과 1을 실행
lda.1        batemapbase,x  ; 픽셀을 갖는 워드를 얻음
and          pbittabn.y     ; 이전의 픽셀칼라마스크 실행
sta          pmask
and          pbittab,y      ; 픽셀마스크
ora          pmask          ; 다른 픽셀과 결합
sta.1        bitmapbase, x  ; 비트맵에 기억
; 비트플레인 2와 3을 실행
lda.1        bitmapbase + 16.x
and          pbittabn.y
sta          pmask
lda          mask2
and          pbittab.y
ora          pmask
sta.1        bitmapbase + 16.x
rts
; 픽셀비트마스크쌍의 256워드테이블
pbittab
rept32  : num_col
dw      $8080, $4040, $2020, %1010, $0808, $0404, $0202, $0101
endr
```

```
; 변환된 워드를 갖는 상기 테이블
pbittabn
rept 32 ; num_col
dw    $7f7f, -$4040, -$2020, -$1010, -$808, -$404, -$202, -$101
endr
; 비트플레인 0 및 1(칼라 0 내지 15)에 대한 칼라마스크
mask1tab
dw   $0000, $00ff, $ff00, $ffff, $0000, $00ff, $ff00, $ffff
dw   $0000, $00ff, $ff00, $ffff, $0000, $00ff, $ff00, $ffff
; 비트플레인 2 및 3(칼라 0 내지 15)에 대한 칼라마스크
mask2tab
dw   $0000, $0000, $0000, $0000, $00ff, $00ff, $00ff, $00ff
dw   $ff00, $ff00, $ff00, $ff00, $ffff, $ffff, $ffff, $ffff
col_size equ    Nember_char_rows * 8 * Number_bit_planes
; 문자칼럼 테이블의 개시에 오프셋
pxoftab
temp -   0
     rept  32  ; 문자칼럼의 수
     dw   temp, temp, temp, temp, temp, temp, temp, temp,
temp -   temp + col_size
     endr
; 칼럼아래의 오프셋
pyoftab
temp -   0
     rept 16  ; 문자로(raw)의 수
     dw  temp
     dw  temp+2
     dw  temp+4
     dw  temp+6
     dw  temp+8
     dw  temp+10
     dw  temp+12
     dw  temp+14
 temp -   temp+32
     endr
```

제7도를 다시 참조하면, 플로팅될 픽셀의 위치를 설정하기 위한 스크린상의 X 좌표와 Y 좌표가 플롯 X 레지스터(202)와 플롯 Y 레지스터(204)로 로드된다.(이들 레지스터는 실제로 레지스터블록(76)내의 R1레지스터와 R2레지스터이다).

상기 플롯 X 레지스터(202)로 로드된 플로팅어드레스의 최하위 3개의 비트는 비트플레인바이트내의 어떤 비트가 지정된 X 및 Y 좌표로 기록될 것인가를 설정한다.

누산기(RO)의 내용은 플롯 X레지스터(202)의 최하위 비트들에 의해 선택된 칼라매트릭스회로(206)의 칼럼에 로드된다.

플롯 X레지스터(202)가 000이면 최하위비트는 픽셀을 설정하는 8비트 각각으로 갱신된다.

플롯 X레지스터(202)가 0인 상태에서 3×8디멀티플렉서(212)는 최하위 비트를 세팅하여 비트펜딩 레지스터(210)에 논리 "1" 을 설정시킨다.

RAM제어기(88)는 비트펜딩 레지스터(210)내의 대응하는 비트가 변경이 요구되지 않는다고 지시하기 때문에 비트펜딩레지스터(210)를 이용하여 구지 ROM으로부터 기록될 필요가 없는 점을 지시하지 않는다.

비트펜딩 레지스터(210)는 새로운 데이타를 원하지도 않는데도 RAM으로부터 그 새로운 데이타가 오버라이팅(겹쳐쓰기)되는 것을 막기위하여 픽셀 마스크버퍼의 기능을 갖는다.

이 기능을 수행하기 위하여, 제7도의 비트펜딩레지스터(210)의 내용은 칼 라매트릭스회로(206)에 제공된다.

만약 비트펜딩레지스터(210)의 내용이 000 되면 픽셀의 스크린어드레스가 계산되어 플롯어드레스

레지스터(216)로 로드되며, 그리고 이 바이트내의 픽셀위치(어드레스)는 비트펜딩 레지스터(210)내에 동일 비트를 설정하는데 사용된다.

그러나 비트펜딩 레지스터(210)의 내용이 0이 아니라면 BUSY플래그가 세트 된다.

새로이 계산된 어드레스가 플롯어드레스 레지스터(216)의 내용과 동일하다면 그 새로운 픽셀비트 위치 어드레스는 비트펜딩 레지스터내에 설정되고 BUSY 플래그가 리세트된다.

새로운 어드레스가 플롯어드레스 레지스터의 내용과 다르다면 다음 단계가 취해진다 :

1단계. 비트펜딩 레지스터(210)가 FFh를 포함하고 있으면 직접 3단계로 가라.

2단계. RAM내의 어드레스(PLOT_ADDR + Scr.)베이스로부터 임시 데이타버퍼 (PLOT_BUFF)로 바이트를 읽어내라.

3단계. 비트펜딩 레지스터(210)에 의해 마스크된 데이타버퍼 내의 비트 모두가 플롯칼라레지스터어레이의 행 0과 동일하면 5단계로 직접가라.

4단계. 플롯칼라레지스터 어레이의 행 0을 비트펜딩 레지스터에 의해 인에 이블된 PLOT_BUFF내의 모든 비트에 기록시켜라.

데이타버퍼를 RAM의 어드레스(PLOT_ADDR)에 다시 기록하라.

5단계. (PLOT-ADDR+1)와, 플롯칼라레지스터 어레이의 행 1에 동일 동작을 수행하라.

6단계. 8 또는 256 칼라모드이면, (PLOT-ADDR + 16)와, 플롯칼라레지스터 어레이의 행 2에 동일 동작을 수행하라.

모든 칼라비트가 갱신될때까지 계속하라.

플롯 X 레지스터(202)의 내용과 플롯 Y 레지스터(204)의 내용은 제7도의 전가산기 및 반가산기회로로 처리된다.

전가산기(RA) 및 반가산기(HA)의 구성과, 관련논리회로를 제7도에 블록 다이어그램으로 간단히 도시하였다.

어드레스계산은 다음과 같이 행해진다.

어드레스 = scr_base + 2 * y[0...2] + (y[3...7] + x[3...7] * 16 * ((x[3...7] * 4) & & scr_ht)

단, *는 문자사이즈(char_size.)

중간단계는 :

```
                              y7    y6    y5    y4    y3
                        x7    x6    x5    x4    x3    0     0
            x7    x6    x5    x4    x3    0     0     0     0
=============================================================
px9   px8   px7   px6   px5   px4   px3   px2   px1   px0
```

로써, 가령 6개의 전가산기와 4개의 반가산기를 사용하여 10비트의 부분적 결과 px[0...8]를 얻는다.

이 결과는 부분적인 결과를 선택된 스크린모드에 대한 문자사이즈 값으로 제어되는 12 ×3 멀티플렉서에 제공된다.

y의 보다 낮은 하위비트 y[0...2]와 조합된 이것은 16비트 스크린어드레스를 형성한다.

이것은 스크린이 1K경계상에 놓여질 수 있게하는 스크린베이스값 스크린 [9...22]에 더해지는데 이는 어드레스계산을 완료하기 위함이다.

그리고나서, 이 어드레스는 4,16 또는 256칼라해상도가 선택되었는가에 따라 1,2, 또는 4와 어드레스 정보 입력을 곱셈하도록 작동하는 2위치 밸럴시프트회로(214)에 제공된다.

시프트회로(214)의 출력은 RAM어드레스를 기억하는 버퍼기억장치의 역할을 하는 플롯어드레스레지스터(216)에 제공된다.

이 어드레스는 플롯명령이 실행된 후에 플롯 X레지스터(R1)와 플롯 Y 레지스터(R2)의 내용이 변하므로 버퍼될 필요가 있다.

어드레스비교기(218)는 시프트회로(214)의 출력(플롯하드웨어에 의해 결정됨)인 새로운 어드레스와, 플롯어드레스레지스터(216)에 저장된 예전의 어드레스를 비교한다.

비교결과 어드레스가 다르면 새로운 어드레스가 RAM에 기억되어야 한다.

그러나 상기 어드레스비교기(218)는 어드레스레지스터(216)에 저장되어 있는 플롯어드레스(예전의 어드레스)가 시프트회로(214)의 출력(새로운 어드레스)과 같으면 플롯제어기(200)에 제공될 제어신호 PLEQ를 발생시킨다.

상술한 칼라매트릭스회로(206)로 다시 참조하면 이 칼라매트릭스회로(206)는 칼럼씩 판독된다.

비트플레인카운터(208)는 칼라매트릭스회로(206)에 연결되어 어느 칼럼이어야 하는지를 결정한다.

비트플레인카운터(208)는 RAM제어기(88)에 연결되어 RAM동작이 완료될 때 비트플레인카운터(208)의 내용을 증가시키는 신호를 발생시킨다.

칼라매트릭스회로(206)는 제8(b)도에 도시된 것과같은 구성요소의 어레이를 포함하고 있다.

칼라매트릭스회로(206)의 하나의 매트릭스 구성요소에는 64개의 그러한 구성 요소가 있다.

플롯명령이 디코드될때 칼라레지스터(54)의 칼라데이타(COL)가 래치(220)에 로드될 수 있도록 제어기(200)는 명령제어신호(LDPIX)을 래치(220)에 제공한다.

제어기(200)에 의해 제어신호(DUMP)가 발생되었다는 것은 칼라매트릭스(206)내에서의 버퍼링의 제 1레벨이 완료되어 데이타가 스크린상에 출력될 필요가 있음을 나타내는 것이다.

일단 DUMP신호가 발생되면 래치(220)에 저장되어 있는 데이타가 게이트 회로(226)와 래치(228)에 제공된다.

DUMP신호가 게이트회로(226)에 제공되면 이 게이트회로는 그 데이타를 래치 (228)에 제공한다.

동시에, 게이트(224)가 사용중지되어 래치(228)의 비반전 출력으로부터의 피드백루프가 전에 기억된 데이타의 기억상태를 유지못하게 된다.

데이타갭을 메꾸기 위해 RAM으로부터 데이타가 판독될때, 제어신호(BPR)는 제로입력을 게이트(222)에 제공하며 LDRAM신호는 제로상태에 있게 될 것이다.

이 조건하에서, RAM D입력 단자에 제공된 데이타입력은 게이트회로(226)를 통해 래치(228)로 전달된다.

그후 래치(228)내의 데이타는 제7도의 RAM제어기(88)를 통해 RAM데이타 버스상에 실릴 수 있다.

다른 그러한 구성요소는 픽셀데이타를 수퍼 NES문자형식과 호환성이 있는 문자데이타로 변환시킬 수 있도록 조합되어진다.

제9도에 상세하게 도시된 RAM제어기(88)는 게임카트리지 RAM을 액세스하는 것과 관련하여 여러가지 제어신호를 발생시킨다.

카트리지 RAM은 수퍼 NES, 즉 마리오칩내의 플롯하드웨어와, 실행될 마리오칩 프로그램으로부터의 데이타 페치사이에 공유되어야 한다.

RAM제어기(88)는 적절한 어드레스가 적절한 시간에 RAM어드레스버스에 제공되는 것을 보장하는데 도움이 된다.

적절한 시간에 RAM액세스신호를 발생시키는 동작은 제10도의 중재논리 회로(310)로 부분적으로 제어된다.

RAM제어기(88)는 RAM D데이타버스를 통해 RAM데이타핀의 입력과 명령 버스 사이를 멀티플렉스하는 멀티플렉서(304)를 포함하고 있다.

명령버스 또는 RAM데이타버스는 명령 디코더(60)의 출력신호에 응답하여 선택되며, 적절한 RAM출력이 행선지 Z버스상에 놓이게 된다.

또한 RAM제어기(88)는 명령 디코더(60)로부터 수신된 신호의 제어하에서 16비트 X버스 또는 16비트 Y버스로부터 수신된 데이타를 RAM에 라이트시키는데 사용되는 16비트 데이타레지스터를 포함하고 있다.

데이타레지스터에 로드된 데이타는 하위바이트와 상위 바이트로 나뉘어져 멀티플렉서(302), 즉 명령 디코더(60)로부터 수신된 신호에 따라 하위 바이트 또는 상위바이트를 출력시키는 멀티플렉서를 통해 RAM의 데이타핀에 제공된다.

또한 RAM제어기(88)는 20비트 어드레스멀티플렉서(308)를 포함하고 있다.

이 멀티플렉서(308)는 중재회로(310)로부터 수신된 제어신호, 즉 중재회로(310)에서 발생된 코드인식(CACK)신호와 데이타인식(DACK)신호와 플롯인식(PACK)신호로부터 유도된 제어신호에 응답하여 어드레스입력을 선택한다.

멀티플렉서(308)는 수퍼 NES어드레스버스(HA)로부터 어드레스신호를 수신하여 마리오 "오너" 상태 비트가 0으로 세트될때마다 메모리타이밍 신호발생기(312)를 통해 상기 어드레스신호를 RAM어드레스버스에 제공된다.

중재회로(310)는 수신한 신호(RAM)를 통해 마리오칩 RAM오너쉽의 상태를 알수 있으며, 더불어 RAM리프레쉬 제어신호(RFSH)도 수신한다.

이들 RAM신호와 RFSH신호는 제10도의 "중지" 신호를 만들어 내기 위하여 OR게이트에 함께 제공된다.

또한 어드레스멀티플렉서(308)는 16비트 멀티플렉서 레지스터(306)로부터 어드레스입력을 수신한다(제9도).

멀티플렉서 레지스터(306)는 명령 디코더(60)의 선택신호에 따라 Y버스의 내용과 명령버스의 내용중 어느 하나를 수신한다.

또한 어드레스 멀티플렉서(308)는 제9도에 도시된 바와같이 어드레스 입력인 데이타뱅크레지스터(314)의 출력과 프로그램카운터(PC)의 내용을 수신한다.

스크린뱅크레지스터(316)의 출력은 멀티플렉서(308)에 입력될 플롯어드레스 입력의 MSB와, 제7도의 플롯회로로부터 입력될 MSB를 만들어내는데 사용된다.

그리고 주데이타버스(HD)의 데이타가 스크린뱅크레지스터(316)와 데이타 뱅크레지스터(314)에 입력되며, 그리고 주 CPU가 이들 두 레지스터(316)(314)를 어드레싱한다.

제9도에 도시된 바와같이 이들 두 레지스터(314)(316)는 RAM제어기(88) 자체 내에 반드시 내장되는 것이 아니며, 단지 그들의 내용들이 RAM제어기(88)에 제공되는 것이다.

예를들면, 데이타뱅크레지스터(314)는 후술될 ROM제어기(104)내에 내장되며; 스크린뱅크레지스터(316)는 가령 플롯하드웨어(52)내에 내장된다.

어드레스멀티플렉서(308)에 입력될 신호는 다음과 같이 선택된다(제9도).

코드인식신호(CACK)가 발생되면, 코드맹크 및 프로그램카운터(PC)입력이 선택된다.

데이타인식회로(DACK)가 발생되면, 데이타뱅크와 함께 멀티플렉서 레지스터 입력이 선택된다.

플롯인식신호(PACK)가 발생되면 플롯어드레스가 선택된다.

최종적으로 CACK신호 또는 DACK신호 또는 PACK신호가 존재하지 않는다면 주(가령, 수퍼 NES(S NES))어드레스입력(HA)이 선택된다.

멀티플렉서(308)의 20비트 어드레스출력은 이 어드레스신호를 적절한 시간에 RAM(6)(8)에 제공하는 메모리

타이밍 신호발생기(312)에 제공된다.

메모리 타이밍 신호발생기(312)는 중재블록(310)내의 그레이카운터의 출력을 수신한다.

메모리타이밍 신호발생기(312)는 그레이카운터의 출력을 디코드하여, RAM 어드레스버스(RAM A)를 통해 제1도의 RAM(6)(8)을 어드레싱하기 위한 출력 신호를 발생시킨다.

또한, 메모리 타이밍 신호발생기(312)는 제1도에 도시된 바와같이 행어드레스스트로브(RAS)신호, 칼럼어드레스 스트로브(CAS)신호 및 라이트인에이블(WE)신호를 포함하는 RAM(6)(8)액세스용 제어신호를 발생시킨다.

메모리 타이밍 신호발생기(312)는 RAM싸이클이 완료되었음을 제시하기 위하여 중재 논리회로(310)에 피드백되어지는 DONE신호를 발생시킨다.

또한 메모리 타이밍 신호발생기(312), 즉 RAM제어기(88)내에 있는 데이타래치(도시 안됨)에 외부 RAM으로부터 전송되어오는 데이타를 래치시키도록 데이타 래치신호(DATLAT)을 발생시킨다.

그리고나서 그 RAM으로부터 전송되어온 데이타는 마리오칩회로, 가령 RAM데 이타버스(RAME_IN) 에 제공된다.

메모리 타이밍 신호발생기(312)의 RAM A어드레스 신호출력은 게임카트리지 상에 어느 스태틱 RAM에 제공된다.

다이나믹 RAM이 게임카트리지에서 사용되면 제어신호 LES,RAS 및 WE가 발생된다.

스태틱 RAM신호 또는 다이나믹 RAM신호는 상술한 옵션저항기를 세팅하는 것처럼 마리오칩의 구조에 따라 적절하게 발생된다.

제9(a)도에는 메모리 타이밍 신호발생기(312)에 의해 발생된 타이밍신호와,다른 관련신호가 예시적으로 도시되어 있다.

이 제9(a)도에 예시적으로 표기한 어드레스값과 데이타값은 단지 예를들기 위함이다.

RAM DONE신호는 제8(c)도에 도시되어 있다.

RAM어드레싱신호를 적절한 시간에 발생시키는 동작은 중재논리회로(310)에 의해 부분적으로 제어된다.

제10도에 도시된 바와같이, 중재논리회로(310)는 메모리 액세스입력에 관련신호, 즉 캐시 리퀘스트(CACHERQ)신호, 데이타리퀘스트(DATRQ)신호 및 플롯리퀘스트(PLTRQ)신호를 수신한다.

이들 각각의 입력신호들은 래치(325,327,329)에 각각 일시적으로 저장된다.

만일 RAM 또는 ROM에 기억되어 있는 마리오명령을 실행시키고자 할때는, 캐시 RAM에 기억되어 있는 명령을 실행하고자 하는게 아니고 RAM 또는 ROM에 기억되어 있는 명령을 실행코자 한다는 사실을 확인하기 위한 캐시 리퀘스트 신호(CACHERQ)(제10도참조)를 수신하고나서 그 마리오 명령실행이 수행된다.

따라서 이 캐시리퀘스트신호(CACHERQ)는 명령이 CACHE RAM(94)에서 실행될수 없음을, 즉 CACHE RAM(94)에 기억되어 있는 명령이 실행될 수 없음을 나타낸다.

데이타리퀘스트신호(DATARQ)는 RAM액세스를 요구하는 명령(예를들면, 로드 바이트명령, 로드워드명령)을 디코딩함으로써 발생된다.

부가적으로 중재논리회로(310)는 플롯명령을 디코딩하므로써 플롯제어기(200)에 의해 발생된 플롯리퀘스트신호(PLTRQ)를 수신한다.

중재논리회로(310)는 단지 마리오칩이 동작중일때와 마리오 오너 비트가 세트될때에 인에이블될 뿐이며, 이때 상태 레지스터중지(SUSPEND)모드비트가 "0" 상태가 되어 중재논리회로(310)이 인에이블 되었음을 알린다.

3개의 래치들(325,327,329)은 캐시리퀘스트신호, 데이타리퀘스트신호 및 플롯리퀘스트신호를 수신, 기억한 후에 CRQ신호, DRQ신호 및 PRQ신호를 각각 발생 시킨다.

게이트(331,333,335)는 이들 신호를 각 래치의 비반전출력으로부터 수신하여 이들 신호에 우선순위를 설정한다.

즉, 캐시 리퀘스트신호에 최고 우선순위가, 데이타리퀘스트신호에 두번째 최고 우선순위가, 플롯리퀘스트신호에 최저 우선순위가 각각 설정된다.

캐시리퀘스트신호는 이 신호가 캐시 RAM의 명령을 실행하도록하고 그리고 RAM으로부터 명령을 액세스하는 것이 필요함을 나타내기 때문에 최고의 우선 순위 를 할당받는다.

게이트회로(333)(335)는 보다 높은 우선순위를 갖는 리퀘스트신호가 이미 각 래치를 세팅시켰다면 더 낮은 우선순위를 갖는 리퀘스트신호는 래치(339)(341)를 세팅시키지 못한다는 사실을 보장하는데 사용된다.

시스템이 현재 중지모드상태에 있지 않으면 중지모드신호가 각 게이트(331,333,335)에 입력되므로 래치들(337,339,341)은 세팅될 수 있다.

중지모드신호는 마리오칩이 RAM을 자유로이 액세스하고 있을때 저논리레벨 상태에 있게 된다.

인식 래치(337,339,341)중 일부가 이미 논리 "1" (즉, 사이클이 이미 처리중에 있음)상태에 있을때에 중지모드가 논리 "1" 로 세트되면 이들 래치(337,339,341)는 세트될 수 없다.

게이트(331,333,335)는 RAM을 액세스하는 우선순위를 설정한다.

데이타인식 래치(339)는 캐시 리퀘스트래치(337)가 세트되면 세트되지 않으며, 플롯인식 래치(341)도 캐시 리퀘스트래치 또는 데이타리퀘스트래치가 세트되 면 세트되지 않는다.

캐시인식신호(CACK)는 래치(337)가 캐시요구신호에 의해 세트되자마자 그리고 그 래치가 캐시(94)(또는 RAM)가 사용가능한 제10도의 논리회로에 의해 설정하자마자 발생된다.

데이타인식신호(DACK) 및 플롯리퀘스트 인식신호(PACK)는 제10도의 논리회로가 RAM이 달리 분주하지 않음을 결정하면 데이타리퀘스트신호와 플롯 리퀘스트신호를 인식하기 위하여 발생된다.

래치(337,339,341)의 비반전출력은 게이트회로(343)에 연결되고, 이 게이트 회로(343)는 RAM액세스용 타이밍신호를 발생시키는 그레이카운터(345)를 NOR게이트(344)를 통해 리세트시킨다.

한번에 1개의 출력 비트만을 변경시킬 수 있는 그레이 카운터는 RAM의 액세스시간을 제어하는데 편리하게 사용되는 카운터임은 이 분야에서 통상의 지식을 가진자는 잘 알것이다.

NOR게이트(344) 및 래치(337,339,341)는 타이밍신호발생기(312)에 의해 발생된 DONE신호를 수신한다.

DONE신호는 RAM싸이클이 완료되었음을 지시하는 신호이다.

DONE신호가 발생되면 래치되어 왔던 리퀘스트를 클리어하기 위하여 중재논리회로(310)내에 있는 적절한 래치를 클리어시키는 동작이 개시된다.

또한, DONE신호는 RAM액세스가 완료되었음을 지시할 목적으로 캐시제어기(68) 또는 플롯제어기(52)와 같은 회로에 제공된다.

본 발명의 또 다른 실시예에서 마리오칩은 이중클럭시스템을 사용한다.

따라서, 마리오칩프로세서는 전술한 RAM제어기회로가 사용하는 클록을 사용할 필요가 없다.

예를들면 RAM제어기(88)는 수퍼 NES로부터 수신된 21MHz클럭신호로 구동되는데 반해, 마리오칩프로세서는 다른 가변주파수 클럭신호로 구동된다.

따라서 마리오칩프로세서는 동작하는데 21MHz의 클록 레이트에 제한되지 않는다.

이 실시예에서의 마리오칩은 재동기 이중클럭 인터페이싱 기능을 수행키 위한 제11도의 것과 같은 비동기상태머신제어회로를 사용한다.

제11도의 회로는 이 회로가 또 다른 출력레이트에서 동작하는 메모리제어 기 말고 다른 클럭시스템을 사용하여 실행된다는 조건에서 마리오칩프로세서와 인터페이스 하는데 사용된다.

제11도의 재동기회로는 클럭신호(CK)와 동기되지 않은 클럭신호(DIN)를 수신한다.

그 재동기회로는 CK와 동기된 DIN를 사용하여 DIN이 CK보다 주파수가 높은지 낮은지의 여부를 나타내는 신호를 발생시킨다.

제12도에 예를든 바와같이, 제11도의 회로는 신호 DIN에 응답하여 상태 010, 110, 100, 101, 111 그리고 다시 초기상태인 010으로 돌아가는 상태천이를 한다.

제11도의 재동기회로는 ROM제어기(104)와 RAM제어기(88)와 같이 이중클록 신호를 수신하는 인터페이스 회로에 이용된다.

제11도의 재동기회로가 신호 DIN을 수신하면 래치(A)가 게이트(F)에 의해 세팅되기 때문에 상태 "010" 이 상태 "110" 으로 천이된다.

재동기클록(CK)이 로(low)상태로 되자마자 래치(B)가 게이트(E)에 의해 리세트되어 다음 상태 "100" 로 천이된다.

다음에 클럭(CK)이 하이(high)상태로 되면 래치(C)가 게이트(A)에 의해 세트되어 그 다음상태 "101" 로 천이된다.

제11도에 도시된 바와같이 래치(C)의 출력 단자는 Q로 나타내어져 있다.

입력신호가 다시 로상태로 되면 래치(B)가 게이트(C)에 의해 세트되어 다음상태 "111" 로 천이된다.

클럭(CK)이 상태 "111" 후에 다시 로상태로 되면 래치 A가 게이트(G)에 의해 리세트되어 상태 "010" 으로 천이된다.

그후, 클럭(CK)이 다시 하이상태로 되면 래치(C)가 게이트(B)에 의해 리세트 되어 상태머신(state machine)을 초기상태 "010" 으로 복귀시키고, 따라서 출력은 인액티브된다.

제13도에는 제4(b)도의 ROM제어기(104)가 상세하게 도시되어 있다.

이 ROM 제어기(104)는 캐시로더, 즉 ROM(10) 또는 카트리지 RAM에 저장되어 있는 현재 실행될 프로그램 명령을 마리오칩캐시 RAM(94)에 로딩시키는 동작을 부분적으로 제어하는 캐시로더(400)를 포함하고 있다.

그 프로그램명령은 16바이트씩 캐시 RAM(94)에 로드된다.

16바이트 세그먼트의 중간쯤에서 점프명령을 만나더라도 완전한 16바이트 세그먼트가 로딩되기전에는 이 점프명령을 실행하지 말아야 한다.

캐시로더(400)는 2비트 상태머신, 즉 16바이트 캐시세그먼트의 잔여 바이트가 캐시 RAM(94)에 로딩되는 것을 확실하게 보장함으로써 점프명령의 디코딩 동작에 응답하는 2비트 상태머신을 포함하고 있다.

이 2비트 상태머신의 제 1상태는 프로그램이 캐시의 범위밖에서 실행될 것인지 또는 프로그램데이타가 이미 캐시에 로드되어 있는지를 지시하는 아이들(idle) 상태이다.

그리고 제 2상태는 캐시에 로딩시되는 동작과 카트리지 ROM 또는 카트리지 RAM의 명령을 실행하는 동작이 동시에 수행됨을 지시하는 상태이다.

그리고 제 3상태는 16바이트 캐시세그먼트의 모든 바이트가 로드될때까지 사실상 트리거되지 않고 있다가 점프명령이 디코딩됨으로써 트리거된다.

제 4상태는 점프가 실행될때 만나게 되는데, 이 점프는 캐시의 16바이트 경계와 정확하게 일치하지 않는 어드레스부터 시작하며, 이때 그 캐시는 프로그램이 분기되는 어드레스와 일치하는 16바이트 세그먼트의 일부분에 대한 경계에서부터 채워지게 된다.

제4(b)도의 캐시제어기(68)는 캐시신호, 즉 캐시 RAM(94)에 기억되어 있는 원하는 명령을 현재 이용할 수 없음을 지시하는 캐시신호를 출력하여 캐시로더(400)에게 제공해준다.

따라서 명령은 ROM에서 페치되어야만 한다.

코드뱅크신호는 액세스될 어드레스의 최상위 3비트를 식별하여 프로그램 ROM과 RAM중 어느것이 액세스되어야 하는지를 지시한다.

또한, 캐시로더(400)는 프로그램 실행동안 프로그램카운터(PC)의 최하위 비트에 대응하는 계수를 유지하는 카운터(도시안됨)를 포함하고 있다.

이 카운터는 캐시로더(400)의 PC입력을 통해 로드된다.

ROM제어기(104)내의 캐시로더(400)는 마리오프로세서가 어떤 이유로인해 대기상태에 있지않음을 나타내는 WAIT제어신호와, 마리오칩의 "고(GO)" 또는 "진행" 모드에 있음을 나타내는 GO제어신호를 수신한다.

이런 상황하에서, 캐시로더(400)는 코드페치 제어신호를 발생시켜 제13도의 NOR게이트(408)에 제공하고 뒤이어 ROM타이밍 카운터(406)의 클리어입력에 제공한다.

캐시로더(400)가 코드페치신호(CODEFETCH)를 발생시키면, ROM제어기(104)내의 논리회로는 데이타페치에 앞서 코드페치가 개시되어야 하기 때문에 상위 우선 순위에 있는 코드페치를 개시하고나서 데이타페치를 개시한다.

제10도에 함께 도시된 우선순위 논리회로를 포함하는 중재회로는 발생된 신호가 데이타 페치명령보다 더 높은 우선순위를 갖도록하는데 사용된다.

클리어신호가 ROM타이밍카운터(406)로부터 제거되면 계수싸이클이 개시된다.

ROM타이밍카운터(406)는 ROMRDY타이밍신호, 즉 ROM데이타핀에서 ROM데이타를 이용할수 있음을 지시하는 신호를 발생시키는데 사용되며 그리고 이 신호는 게이트회로(410)로부터 출력된다.

ROM데이타 준비신호(ROMRDY)게이트는 재동기회로, 예를들면 제11도의 재동기회로를 포함하고 있는 재동기회로(402)에 연결된다.

프로세서클럭을 사용하여 동기가 얻어진 후 래치(404)를 리세트시키기 위하여 그리고 레지스터(R14)를 액세스(EN_R14신호발생)함으로써 개시되는 데이타페치 동작을 지시하는 DATAFETCH신호를 발생시키기 위하여 ROM DCK 신호를 발생된다.

데이타페치신호는 ROM타이밍카운터(406)가 소정의 계수(ROM 데이타핀에서 데이타가 이용가능함을 보장하는 계수)에 도달할때 발생된다.

제13도의 ROM제어기는 다음 입력들중 하나의 입력으로부터 어드레스 정보를 선택하는 기능을 갖는 멀티플렉서(414)의 출력에 ROM어드레스를 발생시킨다.

코드뱅크레지스터(412)는 어느 ROM프로그램 뱅크로부터 마리오코드가 실행되 어야 하는지를 설정하기 위하여 수퍼 NES데이타버스(HD)로부터 로드된다.

코드뱅크레지스터(412)는 32비트 ROM어드레스의 8비트를 멀티플렉서 (414) 에 제공한다.

ROM어드레스의 최하위비트는 프로그램카운터(PC)의 내용으로부터 제공되어진다.

캐시 RAM에 데이타를 기록하고 있는 중에 캐시로드(CACHE LOAD)신호의 최하위 4비트가 캐시로더(400)에 의해 발생된다.

부가적인 멀티플렉서(414)의 어드레스입력은 레지스터(R14)가 액세스될 때 마다 마리오범용레지스터(R14)의 내용으로부터 발생된다.

레지스터(RI4)를 액세스하면 데이타페치래치(404)는 데이타페치(DATAFETCH) 신호, 즉 멀티플렉서(414)로 하여금 R14입력(및 수퍼 NES데이타버스(HD)로부터 로드된 데이타뱅크레지스터(416)의 내용)을 선택하게 하는 제어신호로서 사용되는 데이타 페치신호를 발생시켰다.

데이타뱅크레지스터(416)는 R14페칭동작과 관련된 데이타뱅크의 최상위 비트를 기억하고 있다.

부가적으로 데이타페치신호는 ROM타이 밍카운터(BO6)의 계수동작을 개시시키고, 뒤이어 게이트(410)를 통해 ROM준비신호(ROMRDY)를 발생시키는 게이트(408)에 제공된다.

ROMRDY신호가 발생되면 ROM데이타버스(ROM D)[7 : 0]상의 데이타를 이용할 수 있다.

또한 어드레스 멀티플렉서(414)는 수퍼 NES어드레스버스(HA)로부터 ROM 어드 레스를 수신한다.

수퍼 NES어드레스버스는 멀티 플렉서(414)의 제어입력에 연결되는 신호 "ROM" 의 상태에 따라 선택된다.

그 "ROM" 제어신호는 마리오 ROM제어기에게 수퍼 NES가 ROM어드레스버스를 제어하고 있음을 지시한다.

점프명령이 디코딩되고 난후 어드레스 멀티플렉서(414)는 프로그램 카운터의 내용에 캐시로더(400)의 카운터에 의해 발생된 4개의 최하위 비트를 더한 결과를 제공받는다.

이것은 점프명령이 디코드되기에 앞서 캐시세그먼트가 르딩중인 16바이트중 나머지로 로딩되는 것을 가능케한다.

멀티플렉서(422)는 ROM데이타핀(ROM D)부터 마리오칩의 행선지 버스(Z)까지 ROM제어기(104)내에 데이타경로를 제공한다.

래치(404)에 의해 발생된 데이타 페치신호와 ROM타이밍 카운터(406)에 의해 발생된 ROMRDY신호는 ROM버퍼(420)의 로딩을 가능케하는 게이트(418)에 제공된다.

ROM데이타버스(ROM D)(7...0)상의 ROM데이타는 ROM버퍼(420)에 로딩된다.

멀티플렉서(422)는 레지스터(R14)를 액세스함으로써 실행개시되는 데이타 자동페치코드, 즉 게이트B와 같은 명령코드를 디코딩한 결과에 따라 하나의 입력을 선택한다.

코드페치 동작이 디코딩되면 ROM제어기(104)는 제15(a)도의 마리오칩의 명령 버스에 명령을 싣는다.

게트 B명령이 디코딩되면 레지스터(420)에 저장되어 있는 버퍼 바이트는 Z버스상에 실리게 된다.

특정한 게트 B명령 동안은 제13도의 멀티플렉서(422)의 한 입력 단자에 연결된 X버스상에 데이타를 프함하고 있다.

행선지 Z버스상의 데이타는 마리오범용레지스터(76)들중 하나의 범용 레지스 터에 로딩된다.

캐시제어기(68)가 제14도에 보다 상세하게 도시되어 있다.

이 캐시제어기(68)는 태그래치(506)를 포함하고 있다.

이 태그래치(506)는 예를들면 설명을 위해 캐시제어기에 내장된 것으로서 간주된 캐시 RAM(94)에 명령이 저장되어 있는지를 지시하는 기능을 가진 64개의 래치들을 포함하고 있다.

태그래치(506)내의 64개 플래그 각각은 캐시 RAM(94)에 저장되어 있는 16비트의 정보와 대응한다.

ROM 또는 RAM의 명령은 실행됨과 동시에 캐시 RAM(94)에 로드된다.

상기한 바와 같이 점프명령이 실행되면 16바이트 세그먼트의 잔여 바이트가 제13도의 ROM제어기(104)와 함께 기술된 캐시로더(400)를 통해 RAM(94)에 로드된 다.

이들 잔여바이트가 로드될때까지 모든 16바이트 세그먼트는 태그래치(506)를 통해 로딩되 어 플래그(flag)될 수 없다.

14비트감산기(502)는 프로그램카운터가 0에서 15까지 계수하는 동작을 완료 하면 반전된 범위 이탈신호를 출력하며, 게이트회로(510)는 ROM제어기가 1바이트를 출력할 준비가 되었음을 지시하는 ROM데이타 준비신호(ROMRDY)를 출력하면 디멀티플렉서(504)에 의해 어드레스된 위치에 있는 태그래치(506)를 설정한다.

캐시베이스레지스터(500)의 출력과 프로그램카운터의 최상위 비트(예를들면, 비트 3-15)는 감산기(502)에 제공되며 이 감산기(502)는 프로그램카운터(PC)로 부터의 어드레스입력이 캐시 RAM캐치의 범위내에 있는지를 결정한다.

감산기(502)는 캐시 RAM어드레스의 최상위비트(MSB)로서 6개의 LSB를 출력하는데 그 중 3개의 LSB는 프로그램카운터(PC)로부터 제공중이다.

범위이탈신호(0/RANGE)는 감산기(502)의 캐리출력신호에서 발생되어 반전된다.

반전된 범위 이탈신호는 이 신호가 하이일때 래치 어레이(506)내의 하나의 래치의 설정을 개시시키는데 사용된다.

래치설정은 디멀티플렉서(5G4)를 통해 출력되는 감산기(502)의 캐시 어드레스에 따라 좌우됨과 동시에, 명령이 출력캐시 RAM어드레스에 대응하는 캐시에저장 되어 있음을 지시하는 캐시 RAM(94)의 16바이트 세그먼트에 대응한다.

태그래치(506)의 출력은 멀티플렉서, 즉 DEMUX(504)의 64개의 선택 라인 출력들중 하나와 대응하여 출력될 하나의 래치신호를 선택하는 멀티플렉서 선택 입력에 기초하여 64개의 태그래치신호들중의 하나를 NOR게이트에 제공하는 멀티플렉서(512)에 제공된다.

NOR게이트(514)의 다른 입력은 원하는 명령을 캐시 RAM(94)에서 찾을 수 없으므로 외부페치가 필요함을 알리는 범위이탈신호이다.

제15(a)도에는 제4(a)도의 ALU제어기/명령디코더(60)의 블록 다이어그램이 도시되어 있다.

제15도에 도시한 바와같이, ALU제어기/명령 디코더(60)는 캐시 RAM(94), ROM 제어기(104) 및 RAM제어기(88)로부터 명령을 수신한다.

이들 마리오칩 구성요소들은 ALU제어기/명령 디코더(60)의 일부가 아니며 단지 설명을 위해 제15도에 도시되어 있다.

멀티플렉서(525)는 캐시 RAM(94)과 ROM제어기(104)와 RAM제어기(88)로부터 출력되는 명령들중 하나를 선택하며, 이 선택된 명령을 파이프라인래치(527)에 입력시킨다.

멀티플렉서(525)가 RAM(또는 ROM)의 명령들중 어떤 명령을 선택하느냐는 코드뱅크레지스터내의 소정의 비트, 예를들면 비트 4의 상태에 따라 좌우된다.

따라서, 코드뱅크레지스터에 로드된 어드레스정보에 따라 ROM(또는 RAM)으로부터의 명령이 디코드된다.

또한, 멀티플렉서(525)는 캐시 제어기(68)에서 출력된 제어신호(CACHE CTL)의 상태에 따라 캐시 RAM(94)의 명령을 선택하는데, 그 제어신호는 실행될 명령이 캐시 RAM(94)의 범위내에 있음을 그리고 적절한 태그비트가 캐시 제어기와 더불어 기술된 바와같이 설정됨을 지시하는 신호이다.

파이프라인래치(527)는 프로그램카운터 인에이블신호(PCEN, HIGH)로 인에이블되면 멀티플렉서(525)로부터 8비트 명령을 수신한다.

그 인에이블신호는 예를들면 어떤 명령이 ROM(또는 RAM)으로부터 페치중DP 있으며 ROM제어기(104)(또는 RAM제어기(88))에 의해 발생된다.

RAM(또는 ROM)으로부터 명령을 페치하는 동작은 1처리사이클 이상을 필요로하기 때문에 명령디코딩 동작은 각 ROM또는 RAM제어기(104)(88)에 의해 발생된 프로그램카운터 인에이블신호(PCEN)에 의해 개시된다.

한편, 캐시 RAM(94)의 명령이 실행되면 프로그램카운터 인에이블신호(PCEN)가 항상 액티브되어 전반적인 프로세서클럭레이트에서 명령이 실행된다.

ROM(10)을 액세스하는데 걸리는 시간은 캐시 RAN(94)(또는 카트리지 RAM)을 액세스하는데 걸리는 시간보다 훨씬 길기때문에 PCEM신호가 대응캐시 RAM(또는 다이나믹(또는 스태틱 RAM)디코딩 인에이블신호보다 ROM액세스에 대해 덜 빈번한 간격으로 발생되는 것이 필요하다.

파이프라인래치(527)에 일시적으로 저장되어 있는 명령은 옵코드(1.2..... 27)를 지시하는 신호를 발생시키기 위해 게이트회로(537,539,541)에 의해 개략적으로 나타내어진 종래 명령 디코딩 회로에 출력된다.

파이프라인래치(527)에 로드된 명령은 또한 룩어헤드 논리 회로(551)에 제공된다.

룩어 헤드 논리회로(551)는 마리오칩 레지스터블록(76)내의 적절한 레지 스터를 선택하는데 쓰이는 옵코드를 미리 디코딩할 것을 지시하는데 사용되는 회로이다.

따라서 옵코드를 디코딩하기에 앞서 실행속도를 최적화하기 위하여 액세스될 필요가 있는 레지스터, 즉 어떤 레지스터가 액세스되어야 하는가가 명령이 요하는 데이타를 고속으로 액세스할 수 있도록 빠르게 결정된

다.

룩어헤드 논리회로(551)는 여러가지 프로그램 디코딩제어플래그뿐만 아니라 명령 옵코드에도 응답한다.

명령 디코딩회로(60)는 상기한 바와같이 대응하는 프리픽스명령이 디코딩되었음을 지시하는 ALT 1 및 ALT 2신호를 발생시키기 위하여 사전에 디코딩된 옵코드에 응답하는 프로그램제어플래그 검출논리회로(543)를 포함하고 있다.

또한 후술될 관련 ALT 1 PRE신호가 플래그검출기 논리회로(543)에 의해 발생된다.

부가적으로 IL신호와 IH신호가 직접데이타를 요하는 명령이 디코딩되었음을 지시하기 위해 발생된다(여기서 L과 H는 각각 하위비트, 상위비트를 나타낸다).

IH플래그와 IL플래그는 직접데이타 관련명령이 옵코드로서 디코딩되지 못하게 한다.

따라서 낫(not) IL($\overline{IL}$)신호와 낫 IH($\overline{IH}$)신호가 파이프라인래치(527)를 인에 이블시킬 필요가 있다.

전술한 바와같이 ALT 1신호와 ALT 2신호는 다음에 발생된 옵코드를 수정 하는데 사용되며 그리고 이들 신호는 이들 신호에 대한 사전의 논의에 따른 출력옵코드를 수정하기 위하여 예를들면 게이트회로(541)에 보여진 바와같이 디코딩 논 리회로(537,539, 541) 에 입력된다.

룩어헤드 논리회로(551)는 사전에 디코딩된 옵코드 및 이전옵코드(예를들면 프리픽스코드 ALT 1 또는 ALT 2)가 디코딩될때 발생된 신호에 기초하여, 레지스터 선택신호를 발생시킨다.

예를들면 프로그램 제어플래그 검출논리회로(543)내에 도시된 바와같이, ALT 1신호가 디코딩논리회로(545)에 의해 디코딩되면 ALT 1 PRE신호가 발생되는데, 이 신호는 프로그램 제어플래그 검출논리회로(543)에 의해 출력되어 NOR게이트(549)를 경유하여 룩어헤드 논리회로(531)에 제공된다.

또한 ALT 1 PRE신호는 ALT 1래치(547)를 세트시킨다.

또 OR게이트(549)는 래치(547)로부터 ALT 1신호를 입력받아 디코딩논리회로 (537,539,541....)에 제공한다.

제15도에 개략적으로 나타낸 룩어헤드 논리회로는 4개의 레지스터 선택 제어비트(XSEL 0, XSEL 1, XSEL 2, XSEL 3)가 어떻게 발생되는지를 설명하고 있다.

이들 4개의 제 어 비트는 실행중인 명령에 의해 사용되는 X버스로 출력될 16개의 레지스터중 하나의 레지스터의 내용을 선택하는, 제17도의 레지스터 제어논리회로(76)와 함께 기술된 멀티플렉서(620)(622)에 연결된다.

따라서, 파이프라인래치(527)로 로드되기에 앞서 명령이 레지스터 선택 비 트(XSEE-UO)를 발생시키는 룩어헤드 디코딩논리요소(529)에 제공되고, 이어서 래치(535)에 래치되어 신호(XSEL 0)로서 출력된다.

래치(535)는 프로그램카운터신호(PCEN)에 의해 인에이블된다.

유사하게 논리회로(531)는 래치(533)에 래치될 XSEL-UI신호를 발생시키며, 래치(533)는 이 신호를 받아 XSEL 1신호를 출력한다.

ALT 1 PRE신호는 룩어헤드논리(551)회로내의 여러 디코딩논리회로(529,531,....)에 제공됨과 동시에 레지스터 제어 논리회로(76)에 의해 선택된 적절한 레지스터를 설정하는데 이용된다.

예를들면, 룩어헤드논리회로(551)에 도시된 바와같이 ALT 1 PRE신호는 XSEL-U1를 발생시키는 논리회로(531)에 제공된 신호들중 하나의 신호이며, 이 XSEL-UI는 래치(533)에 래치되어 래치(533)의 출력단자를 신호 XSEL 1를 통해 출력시킨다.

제15(b)도에 룩어헤드 논리회로(551)의 동작을 설명하기 위한 예로서의 타이 밍 신호가 도시되어 있다.

제15(b)도에 클럭신호(CK)와, 캐시 RAM데이타 어드레싱에 관련된 예로서의 명령옵코드가 도시되어 있다.

그리고, 파이프라인래치(527)가 로드될때와, 명령 디코딩동작이 수행될때와, 레지스터 선택신호가 발생될때와, 레지스터의 정보가 행선지 Z버스상에 실릴때를 지시하는 타이밍신호들이 도시되어 있다.

제15(b)도에 도시된 바와같이 캐시 RAM데이타 옵코드(옵코드 1)는 클럭펄스(CR)의 상승에지 이후 어떤 시간대에서 유효게 될 것이다.

옵코드는 예를들면 제 2클럭펄스의 상승에지까지 파이프라인래치(527)에 저장되며 이 때 옵코드 2는 래치(527)로 로딩된다.

명령디코더(60)는 제18도의 시간대에서 래치(227)의 출력을 수신하자마자 옵코드 1에 대응하는 명령을 디코딩하기 시작한다.

상기한 바와같이 명령을 디코딩한 결과는 ALU(50), 캐시 제어기(68) 및 플롯하드웨어(52)등과 같은 마리오칩 구성요소에 제어신호를 제공한다.

제15도의 룩어헤드논리회로(551)는 옵코드(2)를 디코딩하기전의 시간대에서 신호 XSEE-U를 발생시킴으로써 레지스터 선택디코딩공정을 개시한다.

XSEL-UO신호는 래치(535)에 래치되기전의 디코딩 논리회로(529)의 출력을 나타낸다.

예를들면 XSEL-O래신호는 어떤 시점에서 래치(535)에 의해 출력됨으로써 명령이 필요로하는 데이타는 가능한한 빨리 적절한 버스에 연결되기 위해 명령실행싸이클에서 가능한 일찍 액세스된다.

레지스터 제어논리회로(78)의 일부가 Y버스 및 Z버스 관련레지스터 선택신호를 발생시키기 위해 제16도에 도시되어 있다.

멀티플렉서(604)는 16개 레지스터중 어떤 레지스터가 Z버스상으로부터 기록될 것인지를 선택한다.

멀티플렉서(606)는 어떤 레지스터를Y버스에 제공할 것인가를 결정한다.

멀티플렉서(604)(606)는 4비트 레지스터(600)(602)로부터 각각 입력을 수신 한다.

레지스터(600)(602)는 상술한 "FROM" 및 "TO" 프리픽스명령을 실행하는데 이용된다.

레지스터(600)(602)는 명령버스의 최하위 비트를 레지스터(600)(602)에 제공하도록 동작하는 "TO" 프리

픽스 및 "FROM" 프리픽스를 디코딩 함으로써 각각 인 에이블된다.

레지스터(600)(602)는 상술한 제어플래그를 리세트시키는데 사용되는 명령에 응답하여 클리어 된다.

또, 멀티플렉서(604)(606)는 레지스터찰럭(76)내의 여러 레지스터로부터 입력을 수신한다.

또, 멀티플렉서(604)(606)는 최하위 4비트가 명령행선지 또는 소스레지스터를 설정하는 명령을 실행하기 위하여 명령 버스상의 최하위 비트로부터 입력을 수신한다.

부가적으로, 수퍼 NES어드레스버스로부터의 소정의 최하위 비트는 수퍼 NES 에게 레지스터세트에 접근할 수 있는 경로를 제공하기 위해 멀티플렉서(604)(606) 에 제공된다.

멀티플렉서(604)(606)는 2버스와Y버스에 각각 제공된 레지스터를 선택한다.

제17도에 레지스터블록(76)과, 제4(b)도의 레지스터 제어 논리회로(78)에 내장된 부가적인 레지스터 선택제어 논리회로가 도시되어 있다.

FROMX레지스터(618)는 FROM명령을 디코딩함으로써 발생된 PRGHSET신호에 의해 세트된다.

멀티플렉서(622)의 출력은 제15(a)도에서 발생 된 XSEL 0비트와 XS디 1비트의 상태에 기초하여 발생되어 X버스에 제공된다.

많은 레지스터(RO-R15)와 관련된 특수목적의 기능들이 이제까지 기술되어 왔으므로 여기에서도 반복않겠다.

레지스터(RO-R3)의 출력은 멀티플렉서(608)에 제공되며, 레지스터(R4-R7)의 출력은 멀티플렉서(610)에 제공되고, 레지스터(R8-R11)의 출력은 멀티플렉서(612)에 제공되며, 그리고 레지스터(R12-R15)의 출력은 멀티플렉서(614)에 제공된 다.

각 멀티플렉서(608,610,612,614)의 깊개의 입력중 하나는 제16도의 멀티플렉서(606)로부터 출력된 Y SEL 1 및 Y SEL 0비트에 의해 선택된다.

그리고, 멀티플렉서(608,610,612,614)의 출력들은 멀티플렉서(616)의 입력이된다.

멀티플렉서(616)의 4개의 입력중의 하나는 제16도의 멀트플렉서(606)로 부터 출력된 Y SEL 2비트 및 Y SEL 3비트의 상태에 기초하여 선택된다.

멀티플렉서(616)의 블록은 버퍼 레지스터(617)에 연결되어 있으며, 그리고 버퍼 레지스터의 출력은 Y버스에 연결되어 있다.

레지스터(RO-R15)의 입력을 계속 설명하면, 각 레지스터는 제16도와 더불어 설명된 바와같이 Z SEL(0-3) 의해 선택된 인에이블입력을 가지고 있다.

또한 각 레지스터는 클럭입력(CK)과 데이타입력(DAIA-IN)을 가지고 있다.

여러가지 곱셈 연산에 사용되는 레지스터(R4)는 또한 디스에이블로 비트입 력 및 디스에이블 하이 비트입력과 인에이블로 비트입력 및 인에이블 하이비트입력을 포함하고 있다.

레지스터(R15), 즉 프로그램카운터(PC)는 현재의 16바이트 캐시세그먼트가 캐시 RAM에 적재될때까지 점프동작을 금지시키는 신호(CCHLD)를 제13도의 ROM제어기내의 캐시로더(400)로부터 수신한다.

부가적으로 프로그램카운터는 분기동작이 시작되어 PC에 레지스터(R13)의 내용을 적재하는 것이 가능함을 지시하는 프로그램 루프펜딩신호(LOOPEN)를 명령 디코더로부터 수신한다.

게다가 레지스터(R15)는 파워온리세트신호(RESET)와 루프명령이 실행중일때 프로그램카운터에 레지스터(R13)의 내용을 적재시키는 입력(RN)을 수신한다.

상기한 바와같이 주 비디오게임시스템과 조합된 본 발명의 그래픽 코프로세서는 다양한 특수효과, 예를들면 다각형기초이동체의 회전, 확대 및 축소와 같은 효과를 낳는데 쓰인다.

제18도에는 마리오칩이 디스플레이될 다각형기초이동체의 일부를 생성하기 위하여 어떻게 프로그램되어지는가를 설명하기 위하여 사다리를 작성용 마리오칩 프로그램의 플로우차트가 도시되어 있다.

이러한 다각형작성용 마리오칩 프로그램은 마리오하드웨어가 프로그램을 실행하는 방법에 대한 상세한 설명과 함께 후술된다.

먼저 제18도의 고급플로우차트에 대해 설명한다.

레지스터블록의 특정 레지스터(R1-R15)는 사다리꼴작성에 이용되는 변수에 관련된다(예를들면, 레지스터(RO)는 픽셀 X위치를 저장하며 레지스터(R2)는 픽셀 Y위치라인을 저장하고 레지스터(R7)는 사다리꼴의 높이를 저장한다...).

그 후, 블록(650)으로 나타낸 바와같이 루프카운터가 세트되어 초기 픽셀값이 계산된다.

블록(652)에 나타낸 바와같이 사다리꼴의 수평라인들중 하나의 라인의 길이를 구하는 체크동작이 행해진다.

즉, 라인의 끝점에서 그 라인의 시작점을 감산한 결과치가 음(-)의 값(-VE)이면 루틴이 블록(660)으로 분기된다.

그러나, 라인의 끝점에서 그 라인의 시작점을 감산한 결과치가 라인의 길이가 초과되지 않았음을 나타내는 양(+)의 값이면 루프카운터의 내용이 감소하게 되고(654), 그리고 플롯픽셀명령이 실행되어 적절한 픽셀이 플로팅되게 된다.

그리고 블록(658)에서 루프카운터의 내용이 0인지가 체크된다.

체크결과 루프카운터의 내용이 00이 아니면 블록(654)로 되돌아가는 점프가 실행되어 루프카운터(654)의 내용이 감소하고 그리고 또 다른 픽셀이 플로팅 된다(654).

그러나 루프카운터의 내용이 00이면 좌측다각형측의 X좌표와 우측다각형측의 X좌표가 갱신된다(660).

그 후 사다리꼴의 Y HEIGHT가 감소되고(662), 그 결과가 00이 아니면 루틴은 블록(650(664))으로 다시 분기되어 재실행될 것이고, Y좌표는 다음 주사라인으로 이동하기 위하여 증가될 것이다(665).

Y HEIGHT가 00이 라면 루틴이 완전히 다 실행되어 사다리꼴이 완성될 것이다 (666).

그래픽을 형성하는데 마리오칩 명령세트를 사용하는 방법을 설명하기 위하여, 제18도의 플로우차트를 실시하는 사다리꼴작성용 예제프로그램이 이제 설명된다.

; 사다리꼴 루프작성

```
rx          =       1       ;  x위치 플로팅
ry          =       2       ;  y위치 플로팅
rx1         =       3       ;  상부좌측 x위치
rx1inc      =       4       ;  상부좌측 x위치증가
rx2                 5       ;  상부우측 x위치
rx2inc      =       6       ;  상부우측 x위치증가
rdy         =       7       ;  사다리꼴 y높이
rlen        =      12       ;  루프계수, hline길이

rloop       =      13       ;  루프라벨

hlines
            miwt    rloop, hlines 2   ; hline루프의 시작설정
hlines 1
            mfrom   rx1     ;  x = (rx1) >> 8
            mto     rx
            mhib

            mfrom   rx2
            mhib
            mto     rlen
            msub    rx              ; 길이, rlen = (rx2>>8) - (rx1>>8)
            mbmi    hlines 3        ; rlen < 0이면 hline용 스킵
            mnop
            minc    rlen            ; 항상 1개의 픽셀작성

hlines 2

            mloop

            mplot                   ; hline 작성

hlines 3
            mwith   rx1             ; rx1+ = rx1inc
            madd    rx1inc

            mwith   rx2             ; rx2+ = rx2inc
            madd    rx2inc

            mdec    rdy             ; rdy- = 1

        mbne    hlines1             ; 준비시간반복
        minc    ry                  ; 다음 y장소
```

마리오칩 하드웨어가 프로그램을 실행시키기 위하여 어떻게 동작하는가를 설명하기 위하여 다음 설명은 상기 사다리꼴 형성프로그램에 논점을 둔다.

사다리꼴 형성프로그램을 실행하기전에 주컴퓨터시스템, 예를들면 수퍼 NES는 그 프로그램을 코드뱅크레지스터와 스크린베이스 레지스터에 직접 기억시킨다.

게다가 수퍼 NES는 XEQ어드레스의 하위 바이트를 ROM제어기(104)의 국부 레지스터에 라이트시킨다.

그리고 나서 수퍼 NES는 국부레지스터의 내용과 조합되어 Z버스에 제공된 상위 바이트를 ROM제어기(104)에 라이트한다.

그 후, 마리오칩 프로그램카운터의 역할을하는 레지스터(B15)가 인에이블 된다.

ROM 제어기(104)에 대한 상기 수퍼 NES라이트동작의 하강에지를 검출하자마자 마리오 "고(GO)" 플래그가 세트된다.

프로그램카운터에서 캐시베이스레지스터를 뺀것이 캐시사이즈보다 크다면, 또는 캐시플래그에 프로그램카운터를 곱한것에 캐시 베이스레지스터를 16으로 나눈것을 뺀결과가 0이면, 프로그램카운터의 내용이 ROM(10)으로 전송되어 ROM타이밍 카운터(제13도의 블록(406))가 개시된다.

초기에, 사다리꼴 서브루틴을 실행하기전에 사다리꼴 루프프로그램에 사용되는 변수는 사다리꼴 프로그램리스트의 초기부분에 나타내어져 있는 바와같이 수퍼마리오레지스터에 제공되는데, 예를들면, "플롯 X위치"인 "rx"가 레지스터(R1)에 제공되고 변수 "rloop"가 레지스터(R13)에 제공된다.

이들 레지스터에 변수가 모두 할당되고나면 사다리꼴프로그램이 다음과같이 실행개시된다.

ROM제어기(104)내의 ROM타이밍카운터(406)가 계수 5(약 200 ns)에 도달하면, 실행될 제 1명령 "IWT rloop, hlines 2" 은 ROM데이타버스에서 제4(a)도의 파이프라인레지스터(62)로 로딩된다.

데이타는 캐시 RAM(94)에 동시에 라이트된다.

명령 "IWT rloop, hlines" 를 실행할때에 프로그램카운터는 증가된다.

"IL" 및 "IM" 플래그는 명령스트림의 다음 2바이트가 직접 데이타(immediate data)임을 나타내도록 설정된다.

ROM타이밍카운터(406)가 5에 도달될때 직접데이타(하위 바이트)는 캐시 RAM(94)에 라이트되어 ROM 제어기(104)내의 임시 레지스터에 기억된다.

ROM페치메카니즘은 반복되고 직접데이타의 상위바이트는 하위바이트와 조합되어 Z버스에 전송된다.

레지스터(R13)가 인에이블되고, 이 레지스터(R13)에 루프카운터를 세트시킬 목적으로 Z버스의 내용이 저장된다.

루틴상의 이시점에서 각 명령은 루프명령을 만날때까지 메모리로부터 페치된다 .

명령 "FROM RX1" 을 실행할때에 이 명령코드의 최하위 4비트는 레지스터 제어기(제16도)내의 4비트 "FROM Y" 레지스터(602)로 로딩된다.

부가적으로 RX1(레지스터(R3))의 데이타가 Y버스상에 인에이블되어 16비트 "FROM X" 레지스터(618)에 저장된다.

"TO RX" 명령을 실행할때에는 이 명령코드의 최하위 4비트가 레지스터 제어 기(제16도)의 4비트 "인에이블 Z" 레지스터(600)로 로드된다.

"HIB" 명령은 "FROM X" 레지스터의 16비트 내용을 X버스상에 실으므로서 실행된다.

ALU는 X버스의 상위 바이트를 Z버스의 하위바이트에 실어 Z버스의 상위 바이트를 0으로 세트시킨다.

이것은 X위치의 소수부를 제거하고 제 1수평라인의 시작점을 레지스터(RX)(레지스터 R1)에 낭겨놓게 된다.

명령 "FROM RX2" 를 실행할때에도 상기한 "FORM RX1" 명령을 실행할때와 유사한 동작이 수행된다.

"HIB" 명령은 레지스터(RO)(누산기로서 동작하는 디폴트 레지스터)에서 제 1수평라인의 끝점을 떠나는 사다리꼴의 상위우측 X좌표에 대해서 상기한 것과 유사한 동작을 일으킨다.

"RLEN" 명령과 'SUB RX'명령은 라인의 끝점에서 라인의 시작점을 감산함으로써 즉 RLEN(RI2) = RO−RX로써 실행된다.

부호플래그는 에러상태를 나타내는 음(−)의 결과가 있으면 세트된다.

"BNI HLINES3" 명령은 부호플래그가 세트되면 제 1바이트가 플래그를 설정하는 2바이트 명령이다.

상태플래그가 세트되면 제 2바이트는 분기오프셋이다(여기에서 R15는 R15에 명령을 더한것과 같다).

상태플래그가 세트되지 않으면, B15는 불변상태이고 보통프로그램 실행이 계속된다.

그리고 "INC RLEN" 명령이 실행된 결과 라인길이 레지스터는 적어도 1개의 픽셀이 플로팅되는 것을 보장하기 위하여 그것에 더해진 것을 갖는다.

"루프" 명령은 R12=R12−1의 계산을 실행시킨다.

R12가 0이 아니면 R15(프로그램카운터)에 R13의 내용이 로딩되어 점프를 일으킨다.

이 시점에 있는 프로그램이 캐시 RAM(94)의 범위내에 있으면 캐시로더 (400)는 점프를 검출하여 캐시 RAM(94)의 중지실행을 그대로 계속 로딩할 것이다.

그것이 완료되면 프로그램카운터에는 새로운 값이 로드되어 다음명령이 캐시 RAM(94)으로부터 페치된다.

"PLOT" 명령을 실행하기 위하여 루프/플롯명령쌍을 수평 라인 작성알고니즘을 형성한다.

"PLOT" 명령은 제4(a)도의 "칼라레지스터(54)" 에 기억되어 있는 칼라세트에 R1,R2(X 및 Y좌표르서)에 의해 어드레싱된 스크린픽셀을 설정한다.

픽셀을 포함하는 문자의 어드레스는 플롯하드웨어(52)에 의해 계산된다.

새로운 픽셀데이타는 마리오칩이 다른 문자위치 에서의 플로팅동작으로 이동할때까지 문자라인버퍼(칼라 매트릭스)에서 지속된다.

모든 칼라정보가 칼라매트릭스내의 더블버퍼 메카니즘의 제 2레벨로 전송되 면 이 정보는 외부메모리 RAM에 기록된다.

"WITH RX1" 및 "ADD RX2 INC" 명령은 사다리꼴의 좌측 X좌표를 갱신시키기 위하여 실행된다.

유사하게, "WITH RX2" 및 "ADD RX2 INC" 는 사다리꼴의 우측 X좌표를 갱신시킨다.

"DEL RDY", "BNE, Hlines1" 및 "INC RY" 명령은 사다리꼴이 완료될때까지 다음 Y 좌표(다음 스캔라인)상으로 이동시킨다.

다음의 프로그램리스트는 마리오칩이 8비트 X점, Y점 및 Z점의 어레이를 회전시키도록 어떻게 프로그램 되어지는가를 예시하고 있다.

이 루틴은 회전동작을 수행하기 위해 본 발명의 예시적인 실시예에 따른 그래픽코프로세서를 프르그래밍하는 것을 보여주고 있다.

이 루틴의 리스트는 다음과 같다 :

```
회전리스트 :
: 8비트 X, Y, Z 점들의 어레이를 회전
: 레지스터내의 회전매트릭스로
```

```
; rmat1211, rmat2113, rmat2322, rmat3231, rmat0033
; 매트릭스원소는 8비트 부호소수
; ie 127 = 127/128 ≒ 1
;   -128 = -128/128 = -1
; 이들은 레지스터당 2개의 8비트원소로 기억됨

rx          =       1       ; x
ry          =       2       ; y
rz          =       3       ; z
rt          =       4       ; temp
rmat1211            = 5     ; 매트릭스원소 11 및 12
rmat2113            = 6     ; 매트릭스원소 13 및 21
rmat2322            = 7     ; 매트릭스원소 22 및 23
rmat3231            = 8     ; 매트릭스원소 31 및 32
```

```
rmat0033            = 9   : 매트릭스원소 33
routptr             = 10  : 회전된 점의 어드레스

msh_rotpoints8
        mivt    r14.pointsaddr    : 회전될 점의 ROM어드레스
        mivt    r12.numpoints     : 회전될 점들의 갯수
        mivt    routptr.m_rotpnts : 회전된 점의 RAM어드레스
        mcache                    : 캐시어드레스세팅
        mmove   r13,pc : 루프어드레스캐시

mmatrotploop
        mto     rx                : x을 얻음
        mgetb
        mine    r14
        mfrom   rmat1211          : 11
        mto     rt
        mmult   rx                : m11*x

        mto     ry                : get y
        mgetb
        minc    r14
        mfrom   rmat2113          : 21
        mhib
        mmult   ry                m21*y
        mto     rt
        madd    rt

        mto     rz                : get z
        mget b
```

```
minc    rt4
mfrom   rmat 3231       : 31
mmult   rz              : m31*z
madd    rt

madd    r0

mhib

mstb    (routptr)       ; 회전된 x기억
minc    routptr

mfrom   rmat1211        : 12
mhib
mto     rt
mmult   rx              : m12*x

mfrom   rmat2322        : 22
mmult   ry              : m22*y
mto     rt
madd    rt

mfrom   rmat3231        ; 32
mhib
mmult   rz              ; m32*z
madd    rt

madd    r0
mhib
mstb    (routptr)       ; 회전된 y기억
minc    routptr
mfrom   rmat2113        : 13
mto     rt
mmult   rx              : m13*x

mfrom   rmat2322        : 23
mhib
mmult   ry              ; m23*y
mto     rt
madd    rt
mfrom   rmat0033        : 33
mmult   rz              : m33*z
madd    rt

madd    r0
mhib
mstb    (routptr)       ; 회전된 z기억

mloop
minc    routptr
```

제19, 20 및 21도는 주컴퓨터시스템 예를들면, 수퍼 NES와 함께 본 발명의 프로그램 가능그래픽 코프로세서를 사용하여 발생된 일부특수효과를 예시하고 있다.

제19도에 도시된 바와같이 대상체 즉, 헬레곱터의 측면이 묘사되어 있다.

이 도면은 마리오칩을 사용함으로써 발생될 수 있는 고급디스플레이가 정확하게 반영되도록 의도한 것은 아니다.

제20도와 제21도에는 제19도의 헬리곱터의 확대, 회전된 모습이 도시되어 있다.

본 발명의 그래픽프로세서는 고속으로 회전, 스케일링된 다각형 기초대상체를 포함하는 반면, 비디오게임 주처리시스템에 부담을 최소로하는 3차원(또는 기타)특수효과를 낳는데 사용된다.

앞서 본 발명을 상세히 기술하였는데 이는 단지 예를들기 위함이었다.

앞선 실시예들은 바람직한 것인반면 수치적인 값들은 변경내지 수정이 이 분야에서 통상의 지식을 가진자에 의해 행해질 수 있으며, 다음의 특허청구범위도 본 발명의 범주내에서 변경 내지 수정이 가능하다.

청구항 1. 외부메모리시스템(19) 및 외부메모리시스템(19)내의 외부메모리(10)에 적어도 일부가 저장되어

있는 비디오그래픽프로그램을 실행하기 위한 제1 처리장치인 주처리장치(20)을 구비한 비디오게임시스템(19)(20)에 있어서, 상기 외부메모리시스템은 상기 외부메모리시스템을 상기 주처리장치에 접속시키기 위한 적어도 하나의 커넥터(1), 상기 주처리장치에 의해 실행되는 상기 비디오그래픽프로그램의 제1세트의 프로그램명령을 저장하고, 상기 비디오그래픽프로그램의 제2세트의 프로그램명령을 저장하기 위한 외부메모리(10) 및 상기 외부메모리(10)에 연결되어 있고, 사용시는 적어도 하나의 상기 커넥터(1)를 경유하여 상기 주처리장치(20)에 연결되는 상기 제2세트의 프로그램 명령을 실행하기 위한 제2 처리장치(2)를 구비하는 것을 특징으로 하는 비디오게임시스템.

청구항 2. 제1항에 있어서, 상기 주처리장치(20)는 비디오게임시스템의 주처리장치이고, 외부메모리시스템(19)은 비디오게임카트리지(19)에 내장된 것을 특징으로 하는 비디오게임시스템.

청구항 3. 제1항에 있어서, 외부메모리(10)와 제2처리장치(2)에 연결되어 어드레스 정보, 데이타정보 및 제어정보를 전송하기 위한 외부메모리버스, RAM 유닛(6)(8), 상기 RAM 유닛(6)(8)과 상기 제2처리장치(2)에 연결되어 어드레스 정보, 데이타 정보 및 제어정보를 전송하기 위한 RAM 유닛버스 및 상기 제2처리장치와 상기 주처리장치간에 어드레스정보, 데이타정보 및 제어정보를 전송하기 위한 주처리장치버스를 추가로 포함하는 비디오게임시스템.

청구항 4. 제3항에 있어서, 상기 제2처리장치가 상기 외부메모리버스와 상기 RAM 유닛버스(131)중 적어도 하나를 액세스하는 동작을 제어하기 위한 수단을 포함하는 것을 특징으로 하는 비디오게임시스템.

청구항 5. 제1항에 있어서, 상기 제2처리장치(2)는 상기 제2처리장치(2)에 의해 실행될 명령이 저장되어 있는 외부메모리의 위치를 식별하기 위해 주처리장치(20)로부터 어드레스정보(HA)(133)를 수신하기 위한 수단을 포함하는 것을 특징으로 하는 비디오게임 시스템.

청구항 6. 제1항에 있어서, 상기 제2처리장치(2)는 외부메모리(10)에 기억되어 있는 상기 제2 세트의 명령중 적어도 일부를 실행하기 위한 연산논리장치(50)와, 상기 외부메모리(10)에 기억되어 있는 적어도 하나의 디스플레이 관련명령을 실행하기 위한 플로팅회로(52)를 포함하고 있는 것을 특징으로 하는 비디오게임시스템.

청구항 7. 제6항에 있어서, 제1데이타 소스버스(X), 제2데이타 소스버스(Y) 및 데이타 행선지버스(Z)를 추가로 포함하고, 상기 각 버스(X, Y, Z)가 상기 연산논리장치(50)와 상기 플로팅회로(52)에 연결되는 것을 특징으로 하는 비디오게임시스템.

청구항 8. 제1항에 있어서, 상기 제2처리장치(2)가 캐시제어기(68), 상기 캐시제어기(68)에 연결된 캐시메모리(94) 및 상기 캐시메모리에 기억되어 있는 명령을 실행하기 위한 수단(95, 60, 50)을 추가로 포함하는 것을 특징으로 하는 비디오게임시스템.

청구항 9. 제1항에 있어서, 상기 제2처리장치가 다수의 레지스터(76)를 포함하고, 상기 다수의 레지스터(76) 중 소정의 레지스터(R14)의 액세싱에 응답하여 외부 메모리 페치동작을 자동적으로 개시시키기 위한 수단(104)을 추가로 포함하는 것을 특징으로 하는 비디오게임시스템.

청구항 10. 제1항에 있어서, 상기 외부메모리(10)는 프로그램 ROM 이고, 상기 제2처리장치에 연결된 RAM을 추가로 포함하는 것을 특징으로 하는 비디오게임시스템.

청구항 11. 제1항에 있어서, 상기 제2처리장치(2)와 주처리장치(20)는 명령을 병렬로 실행하는 것을 특징으로 하는 비디오게임시스템.

청구항 12. 제1항에 있어서, 상기 제2처리장치(2)는 실행될 명령에 파이프라인처리를 수행하는 수단(62)(60)을 포함하는 것을 특징으로 하는 비디오게임시스템.

청구항 13. 제1항에 있어서, 상기 제2처리장치(2)는 상기 제2세트의 명령을 디코딩하기 위한 수단(60)과, 디코딩될 관련명령의 연산코드를 미리 처리하기 위한 룩어헤드수단(551)을 포함하는 것을 특징으로 하는 비디오게임시스템.

청구항 14. 제1항에 있어서, 비디오게임시스템은 대상체를 디스플레이하기 위한 디스플레이(36)를 포함하고 있고, 상기 제2 세트의 명령은 상기 대상체를 회전시키기 위한 명령을 포함하고 있으며, 상기 제2처리장치(2)는 상기 대상체를 회전시키기 위한 명령을 실행시키기 위한 수단을 포함하는 것을 특징으로 하는 비디오게임시스템.

청구항 15. 텔레비전형 디스플레이(36)를 사용하는 비디오게임시스템(19)(20)에 있어서, 비디오게임프로그램의 명령을 실행하기 위한 제1 처리장치인 게임마이크로프로세서(22)와, 상기 게임마이크로프로세서(22)에 연결되어 상기 게임마이크로프로세서의 제어하에서 영상처리작업을 수행하기 위한 영상처리장치(24)와 상기 비디오게임프로그램을 기억하기 위한 프로그램메모리(10)와, 상기 프로그램메모리(10)에 연결되어 있고, 동작시 상기 게임마이크로프로세서(22)와 연결되어 상기 비디오게임프로그램 명령중 적어도 일부의 명령을 실행하기 위한 프로그램가능한 제2 처리장치(2)를 구비하는 것을 특징으로 하는 텔레비젼형 디스플레이를 사용하는 비디오게임시스템.

청구항 16. 제15항에 있어서, 상기 프로그램메모리(10)와 상기 프로그램가능한 제2처리장치(2)에 연결되어 어드레스정보, 데이타정보 및 제어정보를 전송하기 위한 프로그램메모리버스와, RAM 유닛(6)(8)과, 상기 RAM 유닛(6)(8)과 상기 프로그램가능한 제2 처리장치(2)에 연결되어 어드레스 정보, 데이타 정보 및 제어정보를 전송하기 NOI한 RAM 유닛버스와, 상기 프로그램가능한 제2처리장치(2)와 상기 게임마이크로프로세서(22)사이에서 어드레서 정보, 데이타 정보 및 제어정보를 전송하기 위한 게임마이크로프로세서버스를 추가로 포함하는 것을 특징으로 하는 텔레비전형 디스플레이를 사용하는 비디오게임시스템.

청구항 17. 제15항에 있어서, 상기 프로그램가능한 제2처리장치(2)는 상기 프로그램메모리(10)에 기억되어 있는 상기 제2 부분의 명령 중 적어도 일부를 실행하기 위한 연산논리장치(50)와, 상기 프로그램메모리(10)에 기억되어 있는 적어도 하나의 디스플레이 관련명령을 실행하기 위한 플로팅회로(52)를 포함하는 것을 특징으로 하는 텔레비전형 디스플레이를 사용하는 비디오게임시스템.

청구항 18. 제15항에 있어서, 상기 프로그램가능한 제2처리장치(2)는 캐시제어기(68)와, 상기 캐시제어기(68)에 연결된 캐시메모리(94)와, 상기 캐시메모리에 기억되어 있는 명령을 실행하기 위한 수단(95, 60, 50)을 추가로 구비함으로써 상기 제12 처리장치와 상기 게임마이크로프로세서가 명령을 병렬로